School of Computing and Information Systems□The University of Melbourne COMP90049 Knowledge Technologies, Semester 1 2017 Project 2: R ur tweeps as mad as u think? #analysis report

1. Introduction

This task is to apply different machine learning methods for assessing the sentiment of tweets. Given train tweets file, different classifier will be built and different evaluation metrics will be calculated based on the development text file. And then, I chose one of classifier models which has a better performance to label the test tweets file. This report is mainly focusing on analyzing and comparing different machine learning methods and it also involves a short background of related research.

The data set for feature engineering and training classifier model is 2 tweets text files which contain raw tweets for training and evaluation and 2 label text files. Each tweet in tweets file has a label in the label file with the same tweet id. The test tweets text file is the one which need to be labelled. Three arff files are for Weka.

2. Background of Related Research

Machine learning is widely used to automatically label the tweets especially supervised learning methods. Some researchers classify the tweets through unigram model, a features based model and a tree kernel based model and the last one outperforms other two models [2]. Some researchers use different machine learning methods to classify tweets sentiment with respect to a certain term and analyzing the performance of each method [3]. Some researchers experiment different feature selection methods to figure a better way to select the features [4]. And some analyzed the influence of pre-process the tweets [5].

3. Features Engineering and Machine learning methods

There are four sets of features. One is the given 46 attributes based on mutual information and mutual information4 and Pearson's –squared test named as A. The second one consists of all the tokens retrieved from the train tweets text which has been preprocessed. The method of getting features is bag of words and the feature set named as B. The third one is the bigram collocations which are made up of all the bigram tokens and it is named as C. The bigram features combine two words as one feature and are also based on bag of words. The last one is retrieved through using the show_most_informative_features in the nltk package which is based on information gain named as D.

In this report, I am going to analyze three different machine learning methods, which are Naïve Bayesian Classifier(NB), Decision Trees(DT), and K-Nearest Neighbour(KNN) and compare these three machine learning methods along with SVM.

4. Analyzing within 3 Machine Learning Methods

4.1 Naïve Bayesian

Naïve Bayesian is conditional probability classifier. It uses Bayes` theorem based on the naïve assumptions. The naïve assumption is that there is no dependent relation between the features so all the features are considered as conditionally independent. And the Bayes` theorem is to calculate the probability for each class as shown figure1. So, different kind of feature set will influence the performance of NB.

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Figure1. Bayes` theorem

4.1.1 different number of features

The classifier based on different number of features will have different performance. The classifier with less features will not have enough information to distinguish the tweets. Including too much features means high dimensionality, which may increase the time of the calculation and the complexity. The test result shown as figure2 and figure 3 and it is based on feature set B through my python script:

Number of features	100	200	500	800	1000
Accuracy	56.23%	58.38%	60.50%	59.44%	59.60%

Figure2. the table of Accuracy VS the Number of features

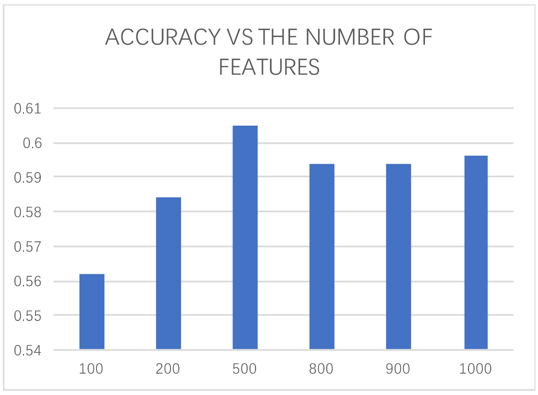


Figure3. the bar chart of Accuracy VS the Number of features

According to the result, the accuracy of the classifier is increased with the number of features smoothly. While the accuracy drops when there are too many features. Large number of features may contain low information features which are common in all the classes. For example, ‘today’ appears in all the classes but it has no contribution to the classifier. Moreover, noisy data like this may lower the performance of the system because it leads to the problem of overfitting.

4.1.2 different structure of features

The classifier based on different structure of the features will have different performance. In the NB, we assume that all the features are independent while it`s not true in text classification. The words in the text are correlate with each other. For instance, terms like ‘not good’ should be considered as one features. If they are separated, the classifier may label this text to be positive based on the term ‘good’ but actually it should be labelled as negative. So, using bigrams as features should have a better performance. The accuracy of these two systems using feature set B and C is shown as figure4 through my python script:

Evaluation metric	Single-word features	Bigram features
Accuracy	60.12%	55.44%

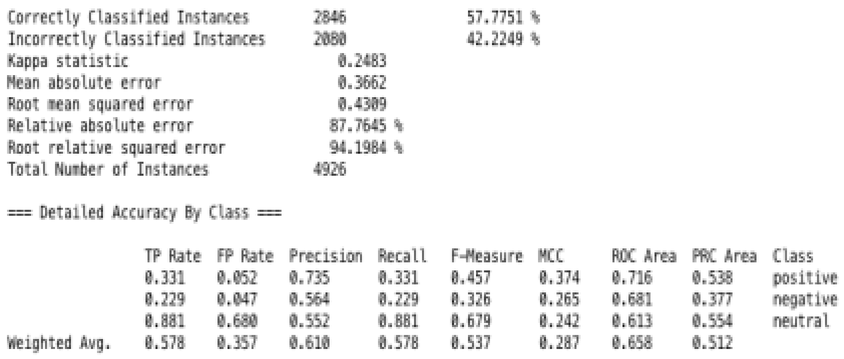
Figure4. the table of Accuracy VS different structure of features

According to the result, the accuracy of the classifier which uses bigrams as features do not have a higher performance as I expected. The reason is that the bigrams are too sparse in the dataset. although using bigrams can smooth the problem of dependency between words, the probability of the text containing this particular bigram is low. So, the accuracy drops.

4.2 Decision Trees

Decision trees is a rule based classification. It builds a decision tree based on the rules in the training dataset. In Weka, DT using C4.5 algorithm which is based on information gain.

4.2.1 Effectiveness of DT



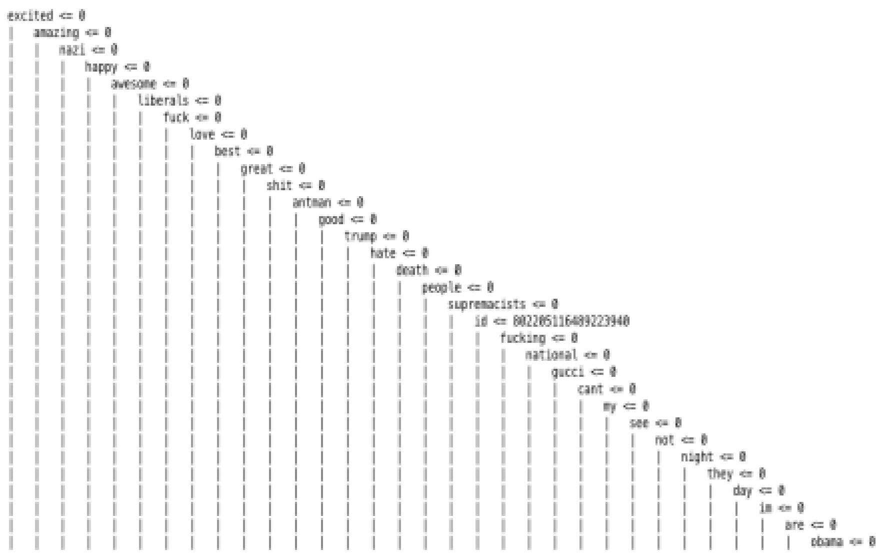
```
Correctly Classified Instances        2846               57.7751 %
Incorrectly Classified Instances      2080               42.2249 %
Kappa statistic                          0.2483
Mean absolute error                      0.3662
Root mean squared error                  0.4309
Relative absolute error                 87.7645 %
Root relative squared error             94.1984 %
Total Number of Instances             4926

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.331    0.052    0.735      0.331    0.457      0.374    0.716     0.538     positive
                 0.229    0.047    0.564      0.229    0.326      0.265    0.681     0.377     negative
                 0.881    0.680    0.552      0.881    0.679      0.242    0.613     0.554     neutral
Weighted Avg.    0.578    0.357    0.610      0.578    0.537      0.287    0.658     0.512
```

Figure5. the effectiveness of DT with feature set A

According to the result, the system is more likely to predict a tweet as neutral. And the positive precision is very high while the positive recall is very low.

4.2.2 J48 pruned tree

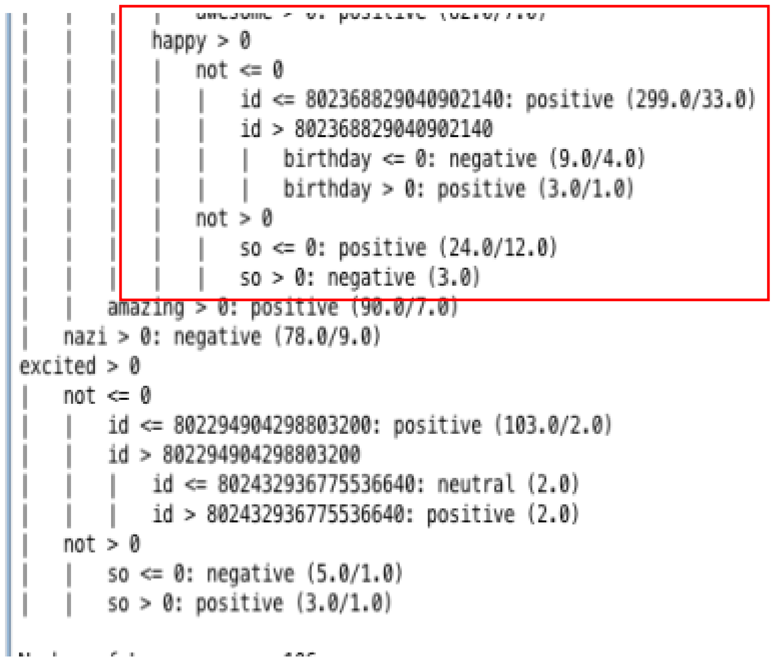
The figures below show some part of the decision tree. As we can see from figure6, most of the features which are related to the sentiment are selected at the top of the tree. These features will have great influence on results.



```
excited <= 0
|   amazing <= 0
|   |   nazi <= 0
|   |   |   happy <= 0
|   |   |   |   awesome <= 0
|   |   |   |   |   liberals <= 0
|   |   |   |   |   |   fuck <= 0
|   |   |   |   |   |   |   love <= 0
|   |   |   |   |   |   |   |   best <= 0
|   |   |   |   |   |   |   |   |   great <= 0
|   |   |   |   |   |   |   |   |   |   shit <= 0
|   |   |   |   |   |   |   |   |   |   |   antman <= 0
|   |   |   |   |   |   |   |   |   |   |   |   good <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   trump <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   hate <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   death <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   people <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   supremacists <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   id <= 802205116489223940
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   fucking <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   national <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   gucci <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   cant <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   my <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   see <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   not <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   night <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   they <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   day <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   im <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   are <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   obama <= 0
```

Figure6. root part of the tree

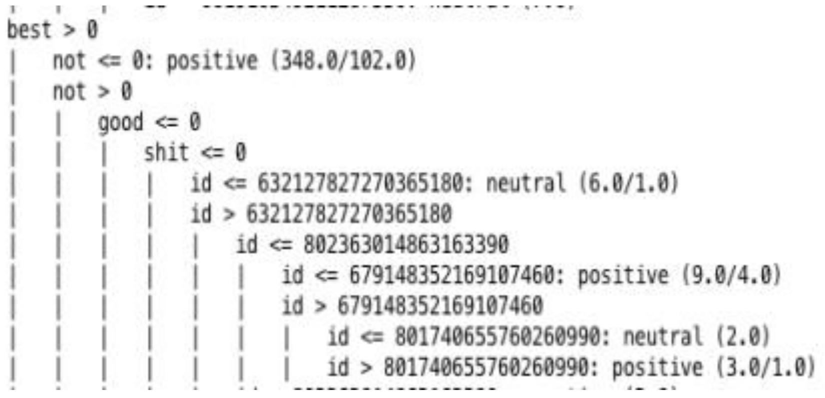
And I notice that terms like ‘not’ or ‘cant’ are used frequently in the tree to differ the sentiment which seems right. However, it has a problem as I highlighted in figure7. The tweet contains term ‘not ’and ‘happy’ will be predicted as positive. In this case, the tweet contains term ‘not happy’ will also be treated as positive. The reason is that ‘not’ and ‘good’ are two separated attributes and the train dataset may have many tweets like ‘i`m happy for not going back’ which labelled as positive. Having these two terms does not mean the tweet is positive or negative.



```
|   |   |   |   awesome > 0: positive (82.0/7.0)
|   |   |   happy > 0
|   |   |   |   not <= 0
|   |   |   |   |   id <= 802368829040902140: positive (299.0/33.0)
|   |   |   |   |   id > 802368829040902140
|   |   |   |   |   |   birthday <= 0: negative (9.0/4.0)
|   |   |   |   |   |   birthday > 0: positive (3.0/1.0)
|   |   |   |   not > 0
|   |   |   |   |   so <= 0: positive (24.0/12.0)
|   |   |   |   |   so > 0: negative (3.0)
|   |   amazing > 0: positive (90.0/7.0)
|   nazi > 0: negative (78.0/9.0)
excited > 0
|   not <= 0
|   |   id <= 802294904298803200: positive (103.0/2.0)
|   |   id > 802294904298803200
|   |   |   id <= 802432936775536640: neutral (2.0)
|   |   |   id > 802432936775536640: positive (2.0)
|   not > 0
|   |   so <= 0: negative (5.0/1.0)
|   |   so > 0: positive (3.0/1.0)
```

figure7. part of the tree

Another problem is that feature id is also frequently used in the tree while it has no contribution to the result or even lower the performance.



```
best > 0
|   not <= 0: positive (348.0/102.0)
|   not > 0
|   |   good <= 0
|   |   |   shit <= 0
|   |   |   |   id <= 632127827270365180: neutral (6.0/1.0)
|   |   |   |   id > 632127827270365180
|   |   |   |   |   id <= 802363014863163390
|   |   |   |   |   |   id <= 679148352169107460: positive (9.0/4.0)
|   |   |   |   |   |   id > 679148352169107460
|   |   |   |   |   |   |   id <= 801740655760260990: neutral (2.0)
|   |   |   |   |   |   |   id > 801740655760260990: positive (3.0/1.0)
```

Figure8. another part of the tree

4.3 K-Nearest Neighbour

KNN is a geometric based method. It calculates the distance between the test instance with the train instances and find number of k nearest neighbor. It classifier the test instance due to the majority of neighbors` label.

4.3.1 Different value of k

The classifier with different value of k will have different performance. With k increasing, the neighbor area becomes bigger and more instances will be considered as a basis for classification. In this case, we can reduce the influence of outliers. For example, if the system only searches one neighbor which happens to be the outliers. Then the system will label the instance wrongly. Certainly, k cannot become too big. Because more noisy points will be included in the neighbor area which will reduce the performance. So, I test different value of k in Weka, choosing IBk as classifier model with feature set A and the accuracy are shown in figure9 and 10:

k	1	5	10	30	50	60	100
accuracy	47.67%	51.12%	55.34%	57.55%	57.81%	57.71%	57.13%

Figure9. the table of Accuracy VS the Value of K

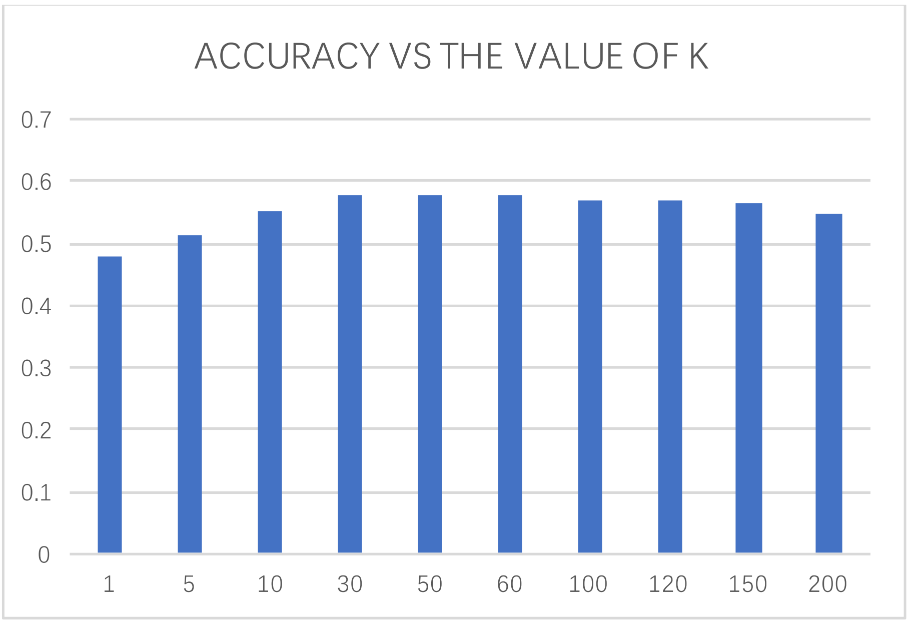


Figure10. the bar chart of the table of Accuracy VS the Value of K

As I expected, the accuracy grows with the value of k but drop down when the value becomes too bigger.

5. Analyzing between 4 machine learning methods

The feature set is A and the system I use is Weka with the default setting.

Evaluation metric	NB	DT(J48)	K-NN(IBk, k=10)	SVM(SMO)
Accuracy	54.41%	57.82%	55.34%	57.99%
Precision(avg)	0.535	0.585	0.546	0.627
Recall (avg)	0.544	0.578	0.553	0.580
F-Measure (avg)	0.534	0.547	0.542	0.527
time	0.19 sec	12.09 sec	0.01 sec	0.07 sec

According to the result, it`s hard to determine which machine learning method is suitable for the text classification. As I have discussed before, some of the methods can be improved by changing the features set or parameter. And the performance of SVM will be enhanced with the number of train-data instances. Apart from that, I notice that DT is more time-consuming to build the decision tree. While it cost a little to classify. KNN does not cost any time due to it does not need to build one but it cost much time on calculating the distance.

6. Conclusion

For text classification, all the machine learning methods I have discussed in this report have their own cons and pros. Through feature engineering to change the features set or changing the parameter, the performance of different methods will be different though the difference may not be obvious. Furthermore, I think the number of training instances may also have influence on the performance. In this report, the number of training instances are all the same.

7. Reference

[1] Rosenthal, Sara, Noura Farra, and Preslav Nakov (2017). SemEval-2017 Task 4: Senti- ment Analysis in Twitter. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval '17). Vancouver, Canada.

[2] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau(n.d). Sentiment Analysis of Twitter Data - Columbia University. Retrieved May 23, 2017, from http://www.cs.brandeis.edu/~cs140b/CS140b_docs/AnnotationPapers/TwitterColumbia.pdf

[3] Alec Go, Lei Huang, and Richa Bhayani. (2009, June 6). Twitter Sentiment Analysis. Retrieved May 20, 2017, from http://ai2-s2-pdfs.s3.amazonaws.com/483a/f11b2904f8e0f4d53a2ccf65bd98eca63eb1.pdf

[4] Mansour R., Hady M.F.A., Hosam E., Amr H., Ashour A. (2015) Feature Selection for Twitter Sentiment Analysis: An Experimental Study. In: Gelbukh A. (eds) Computational Linguistics and Intelligent Text Processing. CICLing 2015. Lecture Notes in Computer Science, vol 9042. Springer, Cham

[5] Bao Y., Quan C., Wang L., Ren F. (2014) The Role of Pre-processing in Twitter Sentiment Analysis. In: Huang DS., Jo KH., Wang L. (eds) Intelligent Computing Methodologies. ICIC 2014. Lecture Notes in Computer Science, vol 8589. Springer, Cham