COMP30027 Assignment 2 Report

Anonymous

1. Introduction

	Unnamed: 0	id	text	sentiment
0	2	805582613687713000	doctors hit campaign trail as race to medical...	neutral
1	3	637480203497832000	is anybody going to the radio station tomorro...	positive
2	4	641096279930507000	i just found out naruto didn't become the 5th...	neutral
3	5	625730917647126000	"prince george reservist who died saturday ju...	neutral
4	6	633292370906230000	season in the sun versi nirvana rancak gak..s...	positive

Figure 1.1- Head of the training data

Tweets have become a popular way for people to express their altitudes and standpoints. Therefore it is important to quickly identify the sentiments of a specific tweet from so much information of the text. This project aims to build and critically analyse several supervised Machine Learning methods, in order to predict the sentiment of Tweets. In the train set provided, each row in the data file contains a tweet ID, the tweet text and the sentiment for that tweet. The sentiments are categorized to positive, negative and neutral respectively. By applying different processing techniques and models I intend to find the optimum prediction based on the training data provided.

2. Method

2.1 Data Pre-processing

In the first step I have to handle the raw data to make it convenient for further analysis. I read the CSV data files and extract the required information from the training data. Then I remove the noise in the raw texts. I only keep words related with the sentiments. Since a validation file is not given I have to split the training data to two parts: some for training and the other for checking the predicted results. Then I apply two vectorization methods (BoW & TFIDF) on the handled data, and separate the original textual instances into many words.

2.1.1 BoW

BoW (Bag of Words) is a vectorization technique that counts words frequency for each textual instance. All words appear in a training instance are added in the feature vector. Although the entire training data set involves a huge number of vocabularies, each instance on average only contains few words. Hence BoW generates a very sparse matrix.

2.1.2 TFIDF

$$w_{d,t} = f_{d,t} \times \log \frac{N}{f_t}$$

Figure 2.1- The formula of TFIDF

TFIDF (Term Frequency – Inverse Document Frequency) is another approach to vectorize textual data. In the formula above, f(d,t) is the frequency of term t in document d, f(t) is the number of documents containing t, and N is the total number of documents in the collection. A word with higher IDF value is considered to be more informative. Similar to BoW, TFIDF method creates a sparse matrix as well.

Tweet_ID, "if i didnt have you i'd never see the sun. #mtvstars lady gaga", positive

[(51027, 1), (44650, 1), (40410, 1), (43384, 1), (22275, 1), (13438, 1), (20604, 1), ...]

[(51027, 0.17), (44650, 0.09), (40410, 0.23), (43384, 0.29), (22275, 0.22), (13438, 0.46), ...]

Figure 2.2- BoW and TFIDF on the same instance

For example, 51027 is the Term_ID for the word 'you', 44650 is the Term_ID for the word 'the' and so on. Since 'you' appears once, BoW counts 1 time. In TFIDF the weighted score of 'you' becomes 0.17.

2.2 Feature Selection

After the data cleaning there are still significant number of features in the training data, which could cause effects to the efficiency and accuracy of the models. I

import VarianceThreshold to delete the words with very low frequency or the common words that appear in most situations.

2.3 Models Selection

In this assignment I select 0-R baseline model, logistic regression, support vector machine and random forest as the models for machine learning. The functions for these models can be directly imported from the python library. For each model with hyperparameters I make many attempts to obtain the optimal prediction result. I also import GridSearch to help me find the best combination of parameters for models with multiple hyperparameters.

3. Results

3.1 Using BoW

```
Execution time of 0-R Baseline model classifier: 0.012 s
Training accuracy:  0.5795374912403644
Validation accuracy:  0.5834561834561834
Macro Precision: 0.194485
Macro Recall: 0.333333
Macro F1 score: 0.245647
Weighted Precision: 0.340421
Weighted Recall: 0.583456
Weighted F1 score: 0.429972
```

Figure 3.1.1- The result of Baseline model (BoW)

This model simply uses the most frequent sentiment in the training data as the predicted results. It runs quite fast in the aspect of time efficiency. The validation is slightly higher than the training accuracy.

```
Execution time of logistic regression classifier: 0.615 s
Training accuracy:  0.810409632413837
Validation accuracy:  0.6036036036036037
Macro Precision: 0.551143
Macro Recall: 0.432154
Macro F1 score: 0.437308
Weighted Precision: 0.584000
Weighted Recall: 0.608845
Weighted F1 score: 0.555383
```

Figure 3.1.2- The result of Logistic Regression (BoW)

The LR model has good time efficiency as it is done in less than one second. The training accuracy is rather high, however there is a significant difference between the training and validation. Overfitting exits in this condition.

```
Execution time of SVM classifier: 21.459 s
Training accuracy:  0.6234949353379626
Validation accuracy:  0.5993447993447993
Macro Precision: 0.643060
Macro Recall: 0.381768
Macro F1 score: 0.347500
Weighted Precision: 0.629982
Weighted Recall: 0.607043
Weighted F1 score: 0.504081
```

Figure 3.1.3- The result of SVM model (BoW)

SVM model takes the longest time among all the models. Both accuracies reach a satisfiable level and does not have a significant difference.

```
Execution time of Random Forest classifier: 0.75 s
Training accuracy:  0.585462190227432
Validation accuracy:  0.5847665847665848
Macro Precision: 0.413256
Macro Recall: 0.336484
Macro F1 score: 0.253716
Weighted Precision: 0.507428
Weighted Recall: 0.584930
Weighted F1 score: 0.436165
```

Figure 3.1.4- The result of Random Forest (BoW)

The Random Forest takes times longer than the other model except SVM. It does not have a high accuracy in neither training nor validation. However there is almost no difference between the two accuracies. The training accuracy might be more reliable than the other models.

3.2 Using TFIDF

```
Execution time of 0-R Baseline model classifier: 0.014 s
Training accuracy:  0.5796011976810855
Validation accuracy:  0.5832923832923833
Macro Precision: 0.194431
Macro Recall: 0.333333
Macro F1 score: 0.245603
Weighted Precision: 0.340230
Weighted Recall: 0.583292
Weighted F1 score: 0.429775
```

Figure 3.2.1- The result of Baseline model (TFIDF)

This model has very similar result to the baseline model using BoW vectorization. It can be assumed that the vectorization method does not impact the baseline model.

```
Execution time of logistic regression classifier: 0.256 s
Training accuracy:  0.68943110148436
Validation accuracy:  0.6422604422604422
Macro Precision: 0.623708
Macro Recall: 0.513631
Macro F1 score: 0.536931
Weighted Precision: 0.640205
Weighted Recall: 0.650123
Weighted F1 score: 0.621985
```

Figure 3.2.2- The result of Logistic Regression (TFIDF)

The LR model using TFIDF has better behaviour than the BoW model. It has advantages in all numeric index except the training accuracy. Nevertheless the overfitting problem in BoW LR model seems to be fixed here as the difference between training accuracy and validation accuracy becomes much smaller.

```
Execution time of SVM classifier: 37.814 s
Training accuracy:  0.6006880295597885
Validation accuracy:  0.5854217854217854
Macro Precision: 0.636986
Macro Recall: 0.374669
Macro F1 score: 0.326955
Weighted Precision: 0.635876
Weighted Recall: 0.609992
Weighted F1 score: 0.493783
```

Figure 3.2.3- The result of SVM model (TFIDF)

Using TFIDF vectorization the SVM model even takes longer to run. Both accuracies decreases compared to the BoW+SVM model.

```
Execution time of Random Forest classifier: 1.373 s
Training accuracy:  0.5901764668407976
Validation accuracy:  0.5906633906633906
Macro Precision: 0.811972
Macro Recall: 0.346160
Macro F1 score: 0.273080
Weighted Precision: 0.722225
Weighted Recall: 0.591810
Weighted F1 score: 0.451233
```

Figure 3.2.4- The result of Random Forest (TFIDF)

The time efficiency decreases compared to BoW+RF, however the accuracies increases. Meanwhile the precision in this model is significantly higher than the other models.

3.3 Results Summary

Vectorization	Model	Accuracy(Valid)
BoW	Baseline	0.5835
TFIDF	Baseline	0.5833
BoW	LR	0.6036
TFIDF	LR	0.6423
BoW	SVM	0.5993
TFIDF	SVM	0.5854
BoW	RF	0.5848
TFIDF	RF	0.5907

Figure 3.3- All vectorization and model combinations

From the table above the combination of TFIDF vectorization and Logistic Regression model has the highest validation accuracy. Therefore I use this model to predict the testing labels.

4. Discussion

4.1 Data Cleaning

Before vectorization I built a data cleaning function to remove all the noises in the raw data. This step creates a clean text input, which improves both efficiency and convenience for the following machine learning process.

```
#Let's see one example tweet
print(X_train_raw[1])
print(X_train_clean[1])
```

```
 is anybody going to the radio station tomorrow to see shawn? me and my friend may go but we would like to m
ake new friends/meet there (:
anybody go radio station tomorrow shawn friend like new friend meet (:
```

Figure 4.1- A text of before and after data cleaning

4.1.1 Remove Numbers

Since the essential concern is to identify whether a text instance is positive, negative or neutral, numerical data does not contribute to such evaluation.

4.1.2 Remove Stop Words

The stop words here refer to those words with high frequencies, such as ‘the’, ‘a’, ‘is’. These kind of words usually do not have a specific meaning hence can be removed in this case.

4.1.3 Remove Special Characters

The textual instances are from Tweets therefore it is inevitable those data contain many internet languages like ‘http’. In addition a popular way to mention someone on Tweet is by using ‘@’ symbol. These kind of words without any actual meaning can be cleaned.

4.1.4 Standardize Text

‘Don’t’ and ‘do not’ actually has exactly the same meaning, however in the vectorization process they might be identified as two distinct words. In a similar way lemmatization is imported, as ‘talk’ and ‘talking’ refer to the same thing in the concept of words. Those different formats or conditions of words do not help in recognizing the sentiments.

4.2 Split Data

Although the given data files contain both train and test, the testing data does not have the corresponding sentiment labels. Hence in order to check the training outcome I import random hold-out method to split the original training

data. Compared to another split method of K-fold Cross-Validation, Random Hold-out has significant advantage in time efficiency, especially when dealing with such huge amount of data.

To avoid the uneven distribution of data, Random Hold-out has the hyperparameter 'shuffle'. Therefore the validation results do not vary much from the training outcomes.

Another important parameter is the 'test_size', which reflects the ratio between train and validation data. The testing data has 6099 instances while the training data has 21802 instance. The ratio equals to 6099/21802 = 0.2797, hence I take the approximate ratio of 0.28, which gives 6105 validation samples and 15697 training samples.

4.3 Feature Selection

I import VarianceThreshold for feature selection. It works by removing features with low variances. In this case it deletes words with extreme frequencies, either extreme high or extreme low, on the scale of the entire textual input. For example, 'hyperparameter' is an academic vocabulary and not likely to appear in common Tweets. 'We' is a highly used word and can be seen in most Tweets. Neither situations above are correlated to the sentiments and should be deleted.

I have to manually select a threshold for this method. I set the value of threshold to be 0.0001. After BoW vectorization the number of features is 118531. Setting threshold to be 0.001 leaves only 299 features, which is too much less than the original features and might influence the accuracy of the models. The threshold 0.00001 fails to select features, in contrast 0.0001 becomes the most appropriate threshold. I set the same threshold in both BoW and TFIDF models to ensure that vectorization is the only variable.

```
(15697, 299) (15697, 11197)  (15697, 118531)
(6105, 299)  (6105, 11197)   (6105, 118531)
(6099, 299)  (6099, 11197)   (6099, 118531)
```

Figure 4.3- Threshold = 0.001, 0.0001, 0.00001 (BoW)

4.4 Models Evaluation

4.4.1 0-R Baseline Model

The theory behind 0-R baseline model is simply majority voting. The validation accuracy of both BoW+Baseline and TFIDF+Baseline reach an incredible level around 58%. The results indicates that the distribution of the sentiments is quite uneven in the training dataset. From the graph below it is obvious that neutral sentiments take the majority among all the sentiments.

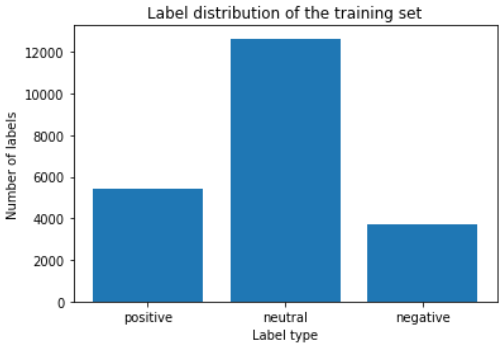

Figure 4.4.1- Label distribution of the training data

4.4.2 Logistic Regression

Logistic regression is easy to interpret, and its output is informative. Meanwhile it is quite efficient when dealing with huge amount of data, which is proved in the execution time in both models with different vectorization.

However the assumption of linearity cannot always be satisfied in logistic regression. This also brings the risk of overfitting. In the Bow LR model, the difference between training accuracy and validation accuracy is over 20%. This data is quite exaggerated and in the actual testing condition the accuracy might decrease even more.

4.4.3 Support Vector Machine

SVM works well with a great number of features. Since with more features the dataset is more likely to be linearly separable.

Nevertheless SVM is not a proper choice when dealing with datasets with too many instances, as the time efficiency will be significantly dragged down. In both BoW and TFIDF vectorization, using SVM take at least 20 seconds to respond. In addition SVM does not calculate the probabilities directly like most

quantitating methods. With so many features it is almost impossible to observe the decision boundary.

An important hyperparameter in SVM is the strength of regularization, which is the value of C in the code. Increasing C means punishment to the slack variable, which makes the slack variable close to 0. In this way the punishment to wrong classifications also increases, resulting in better behaviours in training accuracy. However a high C value leads to overfitting, a low C value treats wrong classification as noises and is less likely to cause overfitting.

```
Training accuracy:  0.8150602025864815
Validation accuracy:  0.5873873873873874

Training accuracy:  0.6234949353379626
Validation accuracy:  0.5993447993447993
```

Figure 4.4.3- C=0.8 (upper), C=0.1 (lower)

4.4.4 Random Forest

The execution of random forest is parallelizable as each random tree is trained independently at the same time. This brings the method good time efficiency. Random forest is like an improved decision tree, as the randomly selected features do not cause overfitting problem like decision trees. However the randomness also makes the decision logic of the model not transparent, as the process of each decision making cannot be observed.

4.5 Evaluation Metrics

Categorizing the sentiments is clearly a classification question. Therefore it is proper to choose macro averaging or weighted averaging, since the micro averaging is actually the same thing as the accuracy in this case.

Macro averaging calculates the precision, recall and F-score for each class, then get the average number. Weighted averaging calculates the weighted average precision, recall and F-score for each class, where the weight is the ratio between each class's instances and the total instances.

The result of baseline model has already indicated that the data distribution is quite uneven, where neutral takes over half of all sentiments. Hence it is reasonable to use weighted averaging as the evaluation metric.

4.6 Error Analysis

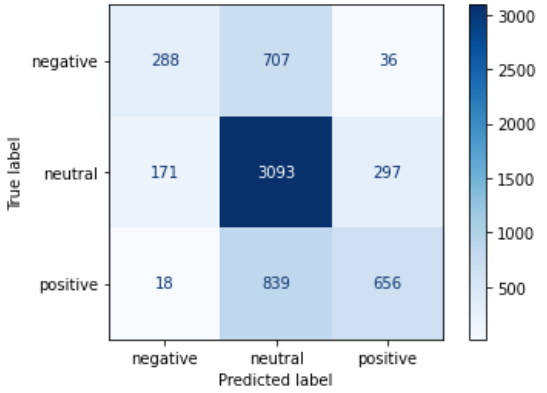

Figure 4.6- Confusion matrix of TFIDF + LR

I plot TFIDF+LR since it has the highest validation accuracy out of all models. Nevertheless the False Negative values for negative and positive texts are rather high. A hypothesis of this condition is that there are too many neutral texts which interrupts the model's classification on negative and positive texts.

Each class has uneven amount of data. To solve this problem I need to balance the amounts between the classes, by adding the positive and negative instances. Or I can just decrease the neutral instances in the training data.

5. Conclusion

In this Tweet sentiment classification project, I clean the raw texts and split the training data. A variety of machine learning models are applied based on the Bow data and TFIDF data respectively. The best experiment result achieves 0.6423 in accuracy.

For further improvements in natural language processing, more advanced technologies will

be required. The new attempts might have better behaviours in sentimental predictions.

6. References

Rosenthal, Sara, Noura Farra, and Preslav Nakov (2017). SemEval-2017 Task 4: sentiment analysis in Twitter. In Proceedings of the 11th International Workshop on semantic evaluation (SemEval '17). Vancouver, Canada.

https://gist.github.com/jiahao87/d57a2535c2ed7315390920ea9296d79f

https://scikit-learn.org/stable/modules/cross_validation.html?highlight=cross_validation#cross-validation-and-model-selection

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highligh

https://blog.csdn.net/TeFuirnever/article/details/99646257

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highli

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforestclassifier#skl