COMP30027 Assignment 2

1 Introduction

Sentiments analysis has been a prominent sector in the Machine Learning department in recent years for its relevance and usefulness in multiple businesses, such as customer service, product development, and even stock trading. Training such a complicated model requires a significant amount of data, and Twitter, being one of the largest database of people's opinion on the Internet, makes the perfect candidate. The data we are using is taken from the 2017 SemEval conference (Rosenthal et al., 2017). This research will discuss the procedures of conducting sentiment analysis, which include preprocessing, choosing suitable Machine Learning models, fine-tuning and applying those models to your dataset, and analyse the result to gain insights of your data and of your training processes.

2 Method

2.1 Preprocess - Feature engineering

2.1.1 Filtering

When choosing features for sentiment analysis, it is ideal to leave out words that do not convey sentiment. Below are the parts of the tweets that we have removed prior to the training process:

- Pronouns, i.e. I, he, she, they, them, etc.
- Hashtags: Hashtags resembles the overall topic of the tweets, the content of the hashtag will remain, only the hash symbol # is removed.
- Stop words, i.e a, the, for, in, etc.
- Punctuation, numbers, and special character.
- Usernames and handles, which comes after the symbol @.
- Hyperlinks, which starts by http or https.

2.1.2 Word stemming

As the English language is vast, encounter a word in the testing data that is not present in the training data seems inevitable. In order to minimize the guess work we have to do when a new word is presented, we will converge words to its most basic form. Lemmatization is the process of find the type of a word and convert it to the most basic form accordingly to the type.

Lemmatization helps decrease the size of the corpus, and allows for better model training as all forms of a word are categorized as one.

2.1.3 Negations handling

There can be a lot of contraction in tweets as it is an informal way of communication. In this dataset we will specifically target negation contractions, as it has an impact on the sentiment of the sentence. Negation handling will find the contracted form of negation, such as "don't", and expand them into its original state, which is "do not".

2.1.4 Characters normalization

Tweets can contain words that have been altered to convey the tone of the writer, or for emphasis purposes (Kumar and Harish, 2020). We decided to convert all characters that appear 3 or more times consecutively in a word to a single character.

2.1.5 Data removal

We noticed that more than half of the dataset is labelled "neutral" with 12,659 instances. On the contrary, only 5428 are positive and 3715 are negative. This can lead to biased towards neutral when performing evaluation. Hence, we decided to cut out half of the neutral tweets in the dataset, to provide a less biased training process.

Additionally, after preprocessing, there can be instances that are empty string, we will also remove those instances to reduce the noise in training data.

2.1.6 Text to features

In order to convert words into features that the Machine Learning models can use, we chose Term frequency - inverse document frequency (TF-IDF) to vectorize the dataset of statements. Although Bag of Words (BoW) is another choice for vectorizing text and both methods do consider each word's total count, TF-IDF also records the frequency that word occurs in any document in the dataset, and use that to offset the count. This make words appears a lot in the corpus and also in a lot of documents have a low point, which prevents overestimation of a word's importance.

2.2 Classifier functions

2.2.1 Zero R - Baseline

Zero R is simply choosing the classifier that appears the most in the training dataset and default to that whenever a testing instance is presented. Due to its simplicity and naive nature, we will use Zero R as our baseline for evaluating the performance of other models.

2.2.2 Naive Bayes

We decided to use Naive Bayes as the first method of classifying the texts. Since the features are discrete, we will use Multinomial Naive Bayes (MNB). We chose Naive Bayes because its nature makes it susceptible to features that appear frequently, in our text analysis those would be the stopwords, or pronouns, so we can test our hypothesis of the importance of preprocessing.

2.2.3 Support Vector Machine

Support Vector Machine (SVM) works by finding a single vector that separates one class from another. In text analysis, each distinct word counts for 1 dimension, therefore the feature space's dimension can be in the range of tens of thousand. This makes SVM ideal in text analysis, as it is robust in high dimensional spaces (Rushdi Saleh et al., 2011), and most of the problem are linearly separable (Joachims, 1998). For the reasons listed above, we will chose Linear SVM as mode of choice, and they are also faster to train, and have less hyper-parameters for fine tuning.

2.3 Evaluation

2.3.1 Evaluation methods

We will implemented 3-fold and 10-fold cross validations on our models, and macro-averaging the performance metrics to find a suitable number of folds for the task.

2.3.2 Evaluation metrics

Here are the list of metrics that we will consider:

- Accuracy
- Precision
- Recall
- F1 score

3 Results

3.1 Zero R

Guessing all instances are neutral yield an classifying accuracy of 0.58, this result is data-specific and not likely to be generalizable.

3.2 Naive Bayes

Naive Bayes without preprocessing can lead to underfitting the model

After applying Multinomial Naive Bayes on the processed dataset, we have the results below, for conciseness, precision, recall and f1-score will be labelled as Pre, Rec and F1, respectively:

Classifier	Pre	Rec	F1
Positive	0.649	0.590	0.618
Negative	0.720	0.200	0.312
Neutral	0.497	0.745	0.596

Average accuracy = 0.561

Table 1: Evaluation metrics of each classifier with MNB using 10-fold cross-validation

Classifier	Pre	Rec	F1
Positive	0.647	0.579	0.610
Negative	0.735	0.179	0.286
Neutral	0.494	0.755	0.597

Average accuracy = 0.556

Table 2: Evaluation metrics of each classifier with MNB using 3-fold cross-validation

3.3 SVM

Unlike MNB, SVM has a hyper-parameter: C - the penalty term for soft-margin SVM, and to increase the performance of the model, we have to fine-tune it. Fine-tuning is the process of finding the optimal value for the hyper-parameter to ensure optimal condition of the training of the model. The process can be done manually by trying out different values, or by solving related mathematical equations to find the optimal solution. We will fine tune our C value manually, on an exponentially increasing sequence, which is the favourable range for C (Hsu et al., 2003), and check the model's accuracy.

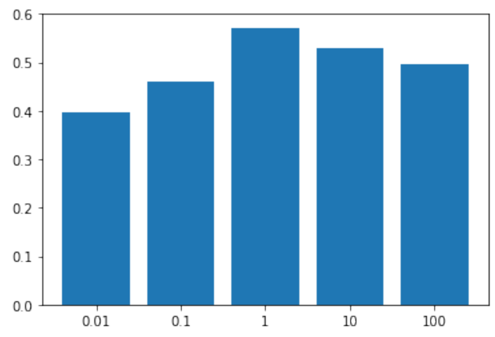

Figure 1: SVM model accuracy with C from 0.01 to 100

The range from 1 to 10 yields the highest accuracy for our SVM model, so we will continue our fine tuning in that range.

Here are the results running with C from 1 to 10:

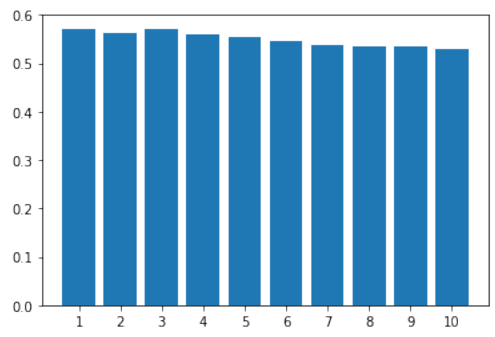

Figure 2: SVM model accuracy with C from 1 to 10

The highest accuracy recorded was 0.572, and the associated C value is 1.

With a fine-tuned model, below are the result of performing SVM on the processed dataset:

Classifier	Pre	Rec	F1
Positive	0.704	0.499	0.583
Negative	0.609	0.476	0.533
Neutral	0.519	0.714	0.600

Average accuracy = 0.583

Table 3: Evaluation metrics of each classifier with SVM on 10-fold cross-validation

Classifier	Pre	Rec	F1
Positive	0.691	0.509	0.582
Negative	0.610	0.456	0.517
Neutral	0.520	0.706	0.596

Average accuracy = 0.577

Table 4: Evaluation metrics of each classifier with SVM on 3-fold cross-validation

4 Discussion

4.1 Selecting fold numbers

10-fold cross-validation typically performed better than 3-fold does, though the difference are not significant. A small fold number indeed offers much quicker training time, larger fold numbers can help even out extreme cases. This is because as the number of folds increases, so does the size of the training data, which helps in averages out extreme cases and outliers. However, too large a fold number can increase training time, and less testing data to evaluate the performance of training model.

4.2 Naive Bayes results

We can see that out of the 3 classes, negative class had the lowest recall. This means that the model rarely ever classify an instance as negative. Part of the reason is the lack of negative instances to train the model. In the dataset there are only 3,715 negative tweets, the lowest of the three sentiments. For the neutral class, the high recall but low precision means that the model have the tendency to make neutral prediction, but is not usually correct when does so. The model has the highest precision of 0.65 in predicting positive class.

To further investigate the reason, we also printed out the 10 highest correlated words with each class:

- Positive: bad, people, say, do, go, like, get, it, trump, not
- Neutral: day, like, see, friday, it, get, tomorrow, go, th, not
- Negative: night, not, it, th, love, happy, go, see, day, tomorrow

Out of the three classes, negative class has the most overlapping features with other classes, making classifying an instance as negative unlikely. These are just the top 10 group, the pattern is likely to repeat for the rest of the corpus. Lack of features that are tightly connected to negativeness can be another reason for the low recall and high precision: if the model classified an instance as negative, it is likely that said instance have features that are exclusive in the class.

Overlapping of informative features decrease the quality of the prediction and especially so in Naive Bayes models. This is why words with high frequency and low importance such as stop words must be left out in the preprocessing.

This is the result of MNB had we omit preprocessing:

Classifier	Pre	Rec	F1
Positive	0.828	0.049	0.092
Negative	0.333	0.001	0.002
Neutral	0.587	0.995	0.736

Average accuracy = 0.590

Table 5: Evaluation metrics of each classifier with SVM on 10-fold cross-validation

The accuracy is only high because the model is very data-specific. Such high bias towards a particular class causes overfitting, and the model would be unable to generalize to new cases.

4.3 SVM results

We will inspect the 10-fold cross-validation results. SVM recorded slightly better results than MNB does, particularly in identifying negative instances. SVM performed relatively well in classifying positive cases, having recorded a precision score of 0.7 and recall of 0.5. Neutral prediction is still favoured, with highest recall of 0.7, but the difference is not as significant as of MNB. SVM model have a recall of 0.496 for predicting negative instances, more than double than that of MNB. This is because SVM does not consider each features as independent individual, it also considers the interaction of features within a class. In literature context, word feature is more than likely to be associated with each other, making the training favourable to SVM.

5 Conclusion

This paper has demonstrated the procedures of conducting a sentiment analysis on text, particularly using Multinomial Naive Bayes and Support Vector Machine and the associated steps. We have found out how the model would behave had we omit preprocessing, and investigate, compare the performance metrics to gain insights of our training process.

References

Chih-wei Hsu, Chih-chung Chang, and Chih-Jen Lin. 2003. A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin. 11.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, pages 137–142, Berlin, Heidelberg. Springer Berlin Heidelberg.

H. M. Keerthi Kumar and B. S. Harish. 2020. A new feature selection method for sentiment analysis in short text. *Journal of Intelligent Systems*, 29(1):1122–1134.

Sara Rosenthal, 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-2017)*, pages 502–518, Vancouver, Canada, August. Association for Computational Linguistics.

M. Rushdi Saleh, M.T. Martín-Valdivia, A. Montejo-Ráez, and L.A. Ureña-López. 2011. Experiments with svm to classify opinions in different domains. *Expert Systems with Applications*, 38(12):14799–14804.