Sentimental Analysis on Tweet

Background

It is clear that commenting on social media (on private accounts and beyond them) has become a new form of written communication as it allows people to avoid being interrupted and express their ideas without any limitations. A technique called Sentiment Analysis analyzes and processes emotions in textual data and specifies whether the given information is positive, negative or neutral about a specific topic or product. The ability to extract insights from social data is a practice that is being widely adopted by organizations across the world. In other words, Sentiment Analysis is competent in understanding the people`s opinion and to provide a basis of evidence and reasoning on particular entities (products, topics, movies, etc..). For example, on an e-commerce site, people can decide to buy a product or not based on reviews from others. Due to the large number of reviews, all reviews can be difficult to read. Therefore, an effective model of Sentiment Analysis for text data saves huge amount of time for human identification and can not only benefit the company well, but also improve customer service.

2. Data and introduction

Human language is very complex and teaching a machine to analyze the various grammatical nuances, cultural variations, slang and misspellings that appear in online mentions is a difficult process. My task is to match each comment with the corresponding sentiment label using several machine-learning based classifiers based on different feature selection methods. I first observe the dataset in this Sentiment Analysis task. The data consists of each comment posted on the social software called "Tweet". Each comment has a corresponding user id. In addition, each comment will be followed by a label reflecting the tone of the comment (positive, neutral, negative). It is obvious that the text is very messy and contains many special characters as well as abbreviations, numbers, slang, apparently misspelled words, and even meaningless words, etc... This certainly adds a lot of difficulties to building models.

season in the sun versi nirvana rancak gak..slow rockkk...	positive
if i didnt have you i'd never see the sun. #mtvstars lady gaga	positive
this is cute. #thisisus @nbcthisisus https://t.co/ndxqyl4gjk	positive
today is the international day for the elimination of violence aga...	neutral
"in his first game back since april 14, david wright went 2-for-5 ...	neutral
josh hamilton flies out to center... we are going to the bottom of...	neutral
watch the trump protest that shut down parts of new york city h...	neutral
wanna see basse try out some dark souls ? see the vid below: d...	neutral
my 8 hour shift will consist of me thinking about ed sheeran to...	positive
seriously debating going to see paper towns on my own tomorr...	neutral
@rafalhill @dumptrump22 they will pay for this on the backs of ...	negative

(Data set)

3. Text pre-processing and feature selection methods

3.1 why text pre-processing and feature selection are important

Sentiment analysis requires text pre-processing and feature selection. Consider the scenario in which you perform an artificial sentiment analysis and discover that the text contains numerous meaningless and distressing words that are useless for sentiment analysis. As a result, text pre-processing is always the initial step in sentiment analysis, as it helps to eliminate certain noisy words and shorten the entire text. Furthermore, feature selection plays a vital role in improving the performance of a predictive model and reducing the modelling computational cost.

3.2 text pre-processing and feature selection methods

For this dataset, my text pre-processing approach includes deleting http links, stop words, converting text to lower case, eliminating punctuation, removing single characters, and so on. The number of unique words drops to 32,625 after text pre-processing, but the data dimension remains large. I used two ways to complete the text vectorization. The first is Bag of Words, which treats each term (word) in the text (training dataset) as an attribute depending on whether it is one of the dataset's most frequently occurring words. Another is TFIDF, which performs similarly to Bag of Words, but return TFIDF value for each term and it reflects how important a word is to a document. (An example is provided below) The rationale for text vectorization is that the input value is text and we need to create features to build models such that every word can be treated as an attribute. I first started my experiment with a feature selection based on the selected common words in the text using Bag of Words method.

4. Models

To show the fundamental performance of predictions without using any models, I started with a zero R baseline. I used cross validation to split the train dataset and test data set based on a limited dataset to make use of and reduce the variance of the whole dataset. After applying 10-fold cross validation, I achieved an average accuracy of 0.581 and the corresponding confusion matrix for each fold. Precision, recall, and f1-score are all omitted from the zero R baseline.

4.1 Multinomial naïve bayes classifier

One of the simple ways to implement text classification is to use Naïve Bayes classifier which is a probability-based classifier and one of advantages to use it is that there is no need to build any extra parameters and it would be implemented really fast. The main idea behind the naïve Bayes is that all features in an instance (text) independently contribute to the probability that belongs to a given class. The multinomial naïve bayes classifier follow the same rule as naïve Bayes and with an assumption that text is represented by the features vectors and each feature is the count in the text. Since our input data has been transformed into vector, we can use the vector to build a frequency table according to the frequent

attribute and make predictions.

4.2 SVM classifier

Instead of using probability-based classifier, I then used SVM. SVM takes data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the classes (red and blue). (See figure 1) This line is the decision boundary: anything that falls to one side of it we will classify as blue, and anything that falls to the other as red. In terms of the SVM we may consider which kernel functions and decision strategy we should use, as well as the choice of regularization parameter C. According to the most sentiment analysis and other similar text classification problems, linear kernel probably would be the first choice to use when there is a lot of features, mapping the data to a higher dimensional space and using another kernel sometimes does not really improve the performance. Decision strategy is another vital point to be considered and I choose one to all strategy since we have multiple classes in this task. I then started with linear kernel and set parameter C to 1.

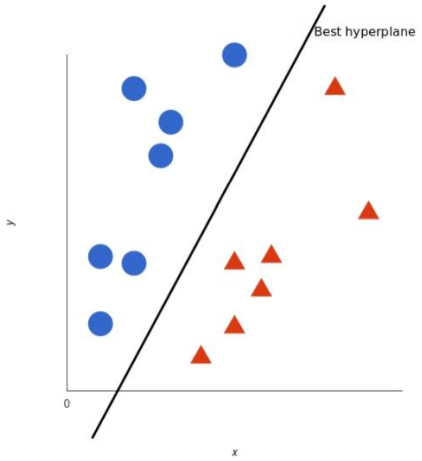

Figure 1

5. Model performance and analysis

In terms of the result, I include the corresponding confusion matrix, macro average scores each fold, as well as the precision and recall for each class, which provide more details and performance between each class. A macro-average will compute the metric independently for each class and then take the average hence treating all classes equally. I use 10-fold cross validation to make use of the whole dataset. All classifiers train models using the same data, which means that if two classifier train model based on fold 1 then both of them using the same data in fold 1.

5.1 Multinomial naïve bayes

In comparison to the zero R baseline, I obtained an average accuracy of 0.609 after applying 10-fold cross validation using a multinomial nave bayes classifier, which is around 3% higher than the zero R baseline. Consider fold 1 (shown below), where accuracy is the ratio of correct positive predictions to total predicted positives, which means how many of the positively

categorized were relevant (tp/(tp+fp)). With 0.693, the neutral class has the highest accuracy, followed by 0.591 and 0.471 for positive and negative classes, respectively. The precision for negative class has the lowest precision and one possible reason for that there are too many false positive cases have been predicted. The macro average precision score for these 3 classes is 0.585 and it would have a bit misleading since the precision of negative class is only 0.471 which is about 11% lower than macro average precision. It is reasonable that 'good' precision for neutral class maintains decent macro average precision. The similar performance can also be found in recall and macro average recall, but recall represents the number of positive class predictions made out of all positive examples in the test dataset. Negative class also has low recall score and we can conclude that the ability of classifying class Negative is pool in this fold. (See figure 2)

```
Fold 1
[[194 171  29]
 [190 847 194]
 [ 28 206 322]]

Classification Report
=====================================================

              precision    recall  f1-score   support

    negative     0.4709    0.4924    0.4814       394
     neutral     0.6920    0.6881    0.6900      1231
    positive     0.5908    0.5791    0.5849       556

    accuracy                         0.6249      2181
   macro avg     0.5846    0.5865    0.5854      2181
weighted avg     0.6263    0.6249    0.6255      2181
```

Figure 2

5.2 SVM

I then implemented SVM. Overall, the average accuracy score is 0.651 which is 5% higher than using multinomial naïve bayes classifier. (See figure 3) In the case of fold 1, the precision in negative class is 0.5894, which is greater than the precision score in 5.1. When we examine the data, we can see that total predicted cases for negative class in two models are different. In 5.1 is 412 (194TP+190FP+28FP), while in 5.2 is 151 (89TP+47FP+15FP) the TP cases and FP cases all decrease which may cause the increase in precision score. In addition, the recall score for negative class decreases sharply with score of 0.2259. These two scores only shows that SVM makes less predictions on negative class. Positive class has the similar behavior that the total predicted cases for this class decrease. However, the accuracy of fold 1 increases and this is because tp and tn cases which are on the main diagonal of the confusion matrix (89 1125 244) increase and the neutral class contribute most of part on the accuracy. In other words, SVM predict more correct neutral class in this fold.

```
Fold 1
[[  89  287   18]
 [  47 1125   59]
 [  15  297  244]]

Classification Report
=====================================================

              precision    recall  f1-score   support

    negative     0.5894    0.2259    0.3266       394
     neutral     0.6583    0.9139    0.7653      1231
    positive     0.7601    0.4388    0.5564       556

    accuracy                         0.6685      2181
   macro avg     0.6693    0.5262    0.5495      2181
weighted avg     0.6718    0.6685    0.6328      2181
```

Figure 3

5.3 Alternations and discoveries

Why these two models show different accuracy scores? Based on the model performance and algorithm behind in 5.1, it is clear that 5.1 relies on each term count and if an input vector has too many 0 counts, then the probability of this term given a class would depends on the choice of alpha in the smoothing method which would affect the prediction result to some extends. For example, we have a vector [1,2,3,0,0] and since there are two 0 counts so the probability of the last two terms would be like (0+1/total number of words in a given class+ total number of unique words in a given class). If the choice of alpha is large and finally the performance of the prediction would be like zero R baseline. However, SVM ignore this and only calculate the equation of a hyperplane to separate classes. Therefore, mathematically speaking, the prediction result using SVM would be more accurate.

Then I try a few other things and tweak the parameters in the models to see if the accuracy can be improved. Instead of changing k frequent words, I used chi-square to select the best k number of features before fitting the model and making predictions because it calculates the importance of the attribute for a given class, whereas selecting k frequent words treats each attribute as having the same weight to the class.

Instead of Bag of Words, I used TFIDF and repeated the same steps, and the overall model performance improved. This is because when TFIDF is used, some words that appear often in specific document but not in the entire document become more relevant, improving the model's performance. (See figure 4 below)

model	frequent word	average accuracy		select k best features	average accuracy		
multinomial naïve bayes	1000	0.6093		1000	0.6325		
	2400	0.6164		2000	0.6273		
	3000	0.6355		3000	0.6291		
				5000	0.6355		BOW
SVM	1000	0.651		1000	0.6581		
	2400	0.6482		2000	0.6642		
	3000	0.6441		3000	0.6624		
				5000	0.6626		
model	frequent word	average accuracy		select k best features	average accuracy		
multinomial naïve bayes	1000	0.6349		1000	0.6325		
	2400	0.6429		2000	0.6273		
	3000	0.6416		3000	0.6291		TFIDF
				5000	0.6355		
SVM	1000	0.6509		1000	0.6531		
	2400	0.6622		2000	0.6577		
	3000	0.6643		3000	0.6592		
				5000	0.6612		

Figure 4

6. Conclusion and potential issues

After carried out several experiment, I found that the model performance of SVM is better than that of multinomial naïve bayes classifier in this task. There are, however, a few concerns that can influence model accuracy. In terms of the dataset itself, it is imbalanced since there are too many occurrences of the neutral class, which accounts for the majority of the data, and the prediction would favor the neutral class. Another difficulty is feature selection, which might be addressed by employing n-grams, in which two consecutive words can be combined to form a single term, which can better predict the tone of the text.

Reference

Go, A. (2009). Sentiment Classification using Distant Supervision. CS224N project report, Stanford.

Rosenthal, S. N. (2017). SemEval-2017 Task4: Sentiment Analysis in Twitter. Proceedings of the 11th.

International Workshop on Semantic Evaluation. Vancouver, Canada: SemEval '17. Schutze, H. M. (2008). Introduction to information retrieval. Cambridge University Press Cambridge.

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.

Hemamalini, Dr.S. Perumal INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 9, ISSUE 04, APRIL 2020. LITERATURE REVIEW ON SENTIMENT ANALYSIS.

https://www.techtarget.com/searchcustomerexperience/tip/Sentiment-analysis-Why-its-necessary-and-how-it-improves-CX
https://aclanthology.org/W04-3253.pdf
https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf