Paper Template for COMP30027 Report

Anonymous

1. Introduction

This report will perform a twitter sentiment analysis on the provided train.csv and predict the sentiments on test.csv and analyse the results based on the methods used.

The train dataset has 21802 instances, with each instance containing id, tweet, and sentiment. Each sentiment contains either one of 'positive', 'neutral', or 'negative'. The test dataset has 6099 instances, with each instance containing id and tweet.

2. Method

Generally, the steps of training machine learning models from given datasets is as follows. Firstly, all the csv files are read into pandas dataframe – this becomes the raw data. The useful information from the data (i.e., instances and features) are then extracted and vectorized, allowing them to be used to train the models. Finally, the performance and quality of the models are evaluated using evaluation metrics such as the accuracy score.

2.1 Feature Engineering

Before using the data to train the models, the features (in this case, words) need to be cleaned first. A significant number of words are irrelevant/low on relevance in predicting the class labels -- they could potentially impact the quality of the models. In addition, the lower number of features also helps with improving the runtime and computational cost of the models.

2.1.1 Text Preprocessing

For each text instance, do the following:

1. Remove all non-alphabetic characters and URLs
2. Remove all non-valid English words
3. Remove all single-character words
4. Stemming

The resulting strings are then vectorized into 'bag of words' vectors.

2.1.2 Feature Selection

The features are further filtered out using the Chi-square feature selection method, with 'k' best features being chosen for further implementation. This is done after the train.csv data is split into train and test sets, to avoid overfitting.

2.2 Resampling

The train.csv data is unbalanced, unlike the test.csv data (more in section 4.5). In order to balance the train data, resampling is required: the number of instances in the non-minority classes are undersampled while the number of instances in the minority class is kept, such that the labels are uniformly distributed.

2.3 Machine Learning

2.3.1 0-R

The baseline model, which is used as a benchmark for the other two models.

2.3.2 Naïve Bayes

Naïve Bayes assumes independence among features. Hence, it allows the computation runtime to be extremely fast while maintaining high accuracy, particularly for high-dimensional features, making it an ideal model to be used in predicting text sentiments.

The resampled train data is randomly shuffled and split such that it will be trained on 80% of the data and tested on the 20% of the data, repeated 5 times for each fold. This cross-validation approach is implemented because it reduces the chance of getting lucky/unlucky with the accuracy and

confusion matrix due to the random nature of the shuffled dataset.

The smoothing parameter 'alpha' is left as default, because: (1) Default is usually the optimum value, there is a reason why such value is chosen as default, and (2) even if it is not the best, it is close to the best that changing the parameter is likely to result in a pyrrhic tiny improvement. Lastly, since the data is uniform, it is obvious the chosen value for the fit_prior parameter is insignificant.

2.3.3 Logistic Regression

Logistic Regression performs particularly well if the labels are linearly separable, which in this case, they are. And unlike Naïve Bayes, it does not make assumptions about the independence of features.

Although this model is computationally slower than Naïve Bayes, the exact same approach and method is used to train Logistic Regression since the runtime is not too slow for the approach to be impractical.

3. Results

Present the results, in terms of evaluation metric(s) and, ideally, illustrative examples and diagrams.

3.3.1 0-R

Since the data is resampled, the labels are distributed uniformly with an accuracy score of 0.333.

3.3.2 Naïve Bayes

This is the average confusion matrix and accuracy for Naïve Bayes using the method mentioned in section 2.

```
average confusion matrix:
[[526.6 130.8  85.6]
 [207.4 299.2 236.4]
 [ 67.2 135.6 540.2]]
average accuracy:  0.6128308658591297
```

The matrix is made using the sklearn library, so the format follows the same definition to that of sklearn's. The top left element is the proportion of true negatives, middle is true neutrals, and the bottom right is true positives.

3.3.3 Logistic Regression

Using the same method, this is the average confusion matrix and accuracy for Logistic Regression.

```
average confusion matrix:
[[483.2 192.8  67. ]
 [169.4 394.2 179.4]
 [ 59.  201.2 482.8]]
average accuracy:  0.6102288021534321
```

4. Discussion/Critical Analysis

4.1 Accuracy Scores

There are multiple possible reasons why both models end up having around 0.61 accuracy scores, but due to word limit, only **one** is explained. One reason is due to the lack of feature data. In the context of analysing text sentiments, a machine learning model generally requires a lot of features, but an instance usually contains very few features. This, and due to undersampling (oversampling can be done instead, but it comes with a whole set of new problems, namely duplicate instances and runtime issues), results in insufficient training instances which could explain the low accuracy scores.

4.2 Vectorization Before Split

In theory, the model should not 'see' any of the test data, so vectorization should be implemented after the split. However, in this case, the data is randomly shuffled, hence the data after the split is a random sample, which should make the vectorization result similar whether it is done before or after the split. Therefore, this minimises the problems that might potentially arise when vectorizing is done before the split, such as overfitting.

Additionally, as explained in 4.1, if the vectorization is done after the split, the lack of feature size becomes more lacking, which could

present a bigger issue than simply vectorizing the words before the split.

4.3 Bag of Words vs TF-IDF

Bag of words (BoW) has an obvious disadvantage where it treats all words as equal despite the obvious fact that some words mean more than others. For example, 'impeccable' (less frequent) has more positive meaning than 'good' (more frequent). However, in the context of text sentiment, whether a word has more meaning than others is not overly significant – both ‘impeccable’ and ‘good’ have positive sentiment, hence the text containing such words is likely to be labelled as positive. Due to the rarity of superlative words such as ‘impeccable’, and the limited number of training instances, they are more prone to being misclassified – and if they are, putting more weight on them can impact the quality of the model’s performance.

Therefore, using BoW is valid in this case, because most words that don't contribute as much to the sentiment (the words that would have been filtered out by the low TF-IDF score) are filtered out using other feature selection methods anyway, such as Chi-square. Ultimately, using either vectorization method is analogous to taking a different road path to reach the same destination, with each path having its own strengths and weaknesses.

4.4 Removing Stop Words

Stop words play an important part in deciding a text’s sentiment. Specifically, only the negation (e.g., not, never) words are important, while the rest are purely noise garbage. Therefore, it can be said that removing all stop words except the negation words would be the best choice here.

So, why are all stop words not removed? The answer is that the result ends up not being too different anyway, since most of the non-negation stop words are likely to be filtered out by Chi-square feature selection. Although in theory, removing non-negation stop words is the ideal choice, in this situation it ends up in an extremely tiny insignificant improvement that it is not worth the sacrifice of computational runtime to remove them.

4.5 Distributions of Train and Test Data

The sentiment labels in the test.csv data is concluded to be uniform, which is not the same as train.csv. The 0-R score is 0.346 for a randomly chosen 40% of the test dataset. Hence, it can be assumed that it has a uniform distribution with close to 100% confidence -- the worst-case scenario where it is the furthest from being uniformly distributed is when the second and third labels are 0.308 and 0.346. Therefore, the full test.csv data should also be uniformly distributed, since the 40% sample is random.

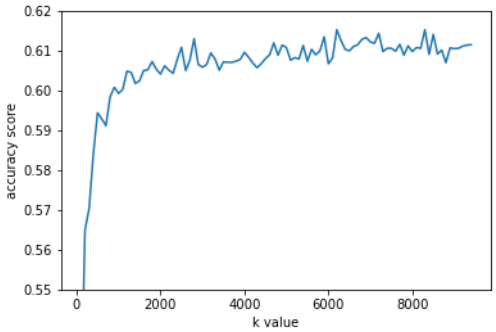
If the train.csv data is not resampled to be uniform, the model’s results are no longer valid, since it is essentially trained on a dataset whose label distribution is different compared to the dataset it is tested on.

4.6 Feature Selection and Model Variance

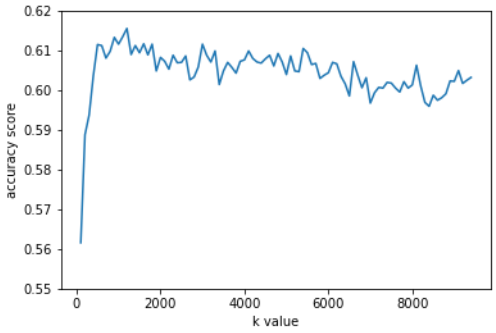
While there are other feature selection methods such as mutual information, Chi-square is the preferred method due to its vastly superior runtime, whereas mutual information’s is too slow to be used in this situation. This does not take away the fact that Chi-square, like other feature selection methods, has its own weaknesses and strengths.

These are the plots of how Chi-square feature selection affects the models’ performance (the number of features selected is denoted as ‘k’ on the x axis).

Naïve Bayes



Logistic Regression



The best value for k between the two models is drastically different. Feature selection has little effect on Naïve Bayes, but the same cannot be said on Logistic Regression.

Accuracy Score	Naïve Bayes	Logistic Regression
Without Chi-square	0.611	0.602
With Chi-square	0.613	0.610

Table 1- Accuracy scores without and with Chi-square feature selection using a close-to-ideal k value, rounded off to three decimal places.

The most likely reason they are different is the "naivety" of Naïve Bayes by assuming the independence among all features – Naïve Bayes performs better on high-dimensional data, because data with a high number of features are generally closer to being independent compared to data with a low number of features. In contrast, Logistic Regression considers the dependence among features. Hence, adding more features in the dataset will only add more noise and, as a result, make it more difficult for the model to predict the correct label.

Another takeaway from the graphs is that although they have different trends, the way both lines move up and down are somewhat similar when k is close to optimum value. A claim can be made that Naïve Bayes has a slightly smaller variance, but the difference is too minuscule for such statement to be confidently concluded. Hence, both models have similar variance.

4.7 Naïve Bayes vs Logistic Regression

In general, Naïve Bayes has (extremely) slightly better accuracy compared to Logistic Regression, with similar variance. However, there is another trend that can be seen between both models.

Naïve Bayes's feature independence assumption is, indeed, naïve. Here is an example where the word 'down' can result in a negative or neutral sentiment, which proves that not all features are independent:

"Playing Messi as a defender brings down the team's performance."

"Louis wrote down his name on the survey."

This means that Naïve Bayes is likely to be biased towards 'positive' and 'negative' sentiments.

Logistic Regression avoids this problem by not assuming feature independence. Although the accuracy of 'neutral' predictions is still not as high as the other two sentiments, it is expected because when a 'neutral' is misclassified, it can go to either 'positive' or 'negative', whereas if a 'positive' is misclassified, it is a lot more likely to go to 'neutral' compared to 'negative' (and the same applies to 'negative'), hence they are less likely to be misclassified.

Ultimately, choosing which model is better is entirely contextual (i.e., whether 'positive' and 'negative' accuracies are preferred to 'neutral'), with each model having their own pros and cons.

5. Conclusions

In conclusion, both Naïve Bayes and Logistic Regression are similar in terms of quality, with each model preforming better in different context. Naïve Bayes has the better ‘positive’ and ‘negative’ prediction, and Logistic Regression has the better ‘neutral’ prediction. Additionally, Naïve Bayes is better when the feature size is big, and the opposite applies for Logistic Regression.

Alas, both model’s accuracies max out at around 0.61, which happens because the given train.csv and test.csv data have different distributions, which ultimately requires train.csv to be undersampled to make the models valid. This, consequently, result in a much lower number of instances and feature size which negatively affects the accuracy score.

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.