Comparison of Classifier Performance for Sentiment Analysis Based on Yelp Dataset

Anonymous

1. Introduction

This report aims to analyse the use of various machine learning approaches and methods on a sentiment analysis task. The task is based on Yelp data of user reviews (Mukherjee et al., 2013 & Rayana and Akoglu, 2015). These reviews will have a 1-star, 3-star, or 5-star ratings which will be the class label in the classification task. As such, this will be a multi-class classification task based on text data.

2. Literature review

Sentiment analysis is one of the most popular and rapidly growing areas of machine learning (Mäntylä, Graziotin and Kuutila, 2018), with applications in wide-ranging fields such as customer insights, financial markets, and political science. Sentiment analysis uses algorithms and machine learning techniques to analyse text, aiming to determine whether the writer has positive or negative (or neutral) views toward a certain topic (Pranckevičius and Marcinkevičius, 2017). In doing this, there are several decisions to be made that are crucial to the effectiveness of the final result.

The first concerns the feature representation of the text. As machine learning models are unable to take text as input, a technique to transform text into numeric features is required. One basic method is bag-of-words, which establishes a vocabulary of words present in the collection of documents and uses the count of words in a document as feature values (Zhou, 2019). Further enhancements to this have been proposed, such as weighting word counts based on their frequency across the entire document collection. This was proposed by Karen Sparck Jones (1972), leading to the popular TF-IDF technique. However, these techniques do not consider word ordering and semantics, with synonyms being as far apart as completely unrelated words, which are limitations overcome by Word2vec and Doc2vec (Le and Mikolov, 2014).

Doc2vec transforms texts into n-dimensional vectors that place similar texts close together in the feature space (Le and Mikolov, 2014). The value of n here can be chosen to suit the specific task and dataset (Nag, 2019), where higher values encode more information but may lead to overfitting and sparse data points. Other attributes may also be added to the Doc2vec features, possibly incorporating information outside the text content.

After choosing a feature representation, a suitable model must be selected. In one sentiment analysis task, da Silva, Hruschka, and Hruschka (2014) found that an ensemble model was the most effective model, followed by Naïve Bayes and Logistic Regression. However, this was a binary classification task based on tweets and different pre-processing methods were used, so their conclusions may not apply to this task.

As such, this investigation is required to identify effective classifiers for the given task, and interpret the relative performance of different classifiers for this problem.

3. Methodology

3.1 Feature Selection

The initial step is feature selection. Three initial machine learning models (Random Forest, Zero-R, and Linear SVC) are used on 3 files (with 50, 100, and 200 Doc2vec features). Each file is split into train and validation sets with a 80/20 split, the models are trained on the training set, and the average

validation accuracy of the 3 classifiers are calculated for each file. The file with 50 Doc2vec features results in the highest average accuracy. Using this file, further feature selection is done. Three features from *review_meta_train.csv* (vote_funny, vote_cool, and vote_useful) are added and the accuracies are recalculated. The addition of these features results in a slightly higher accuracy. Therefore, 53 features are used in the subsequent steps.

3.2 Parameter Tuning

Grid Search with 3-fold cross-validation is used to find the best hyperparameters for Random Forest and SVM. The choice of which hyperparameters to tune focused on those regulating model complexity and regularisation (max_depth, n_estimators, C) and those impacting the optimisation function or selection criteria (criterion, max_iter, penalty). Next, values to try for each hyperparameter are manually chosen, utilising various reasonable values. The set of hyperparameters resulting in the best average performance across all folds are used in further stages of the process.

Due to the randomness involved in the cross-validation process, slight changes in the hyperparameters selected may occur in different runs. The hyperparameters used in subsequent steps reflects those selected in the first run.

3.3 Training-validation Phase

For non-stacking classifiers, models are first fitted using the training set, then used to make predictions for the validation set. Meanwhile, for the stacking classifier, the training set is split into 2 halves to avoid overfitting. 4 different base classifiers (Random Forest, Gaussian Naïve Bayes, Neural Network, and Linear SVC) are first fitted using the first half of the training set. Then, each model is used on the second half of the training set to get the predicted class probabilities or decision function values. This process is repeated, with the base classifiers being trained on the second half of the training set and used on the first half to generate class probabilities and decision function values. These values are attributes for the meta-classifier, which is a Logistic Regression model. A Grid Search with 5-fold cross-validation is used to get the best hyperparameters for the model. Next, the model is fitted with the attributes generated by the base classifiers. In the validation phase, base classifiers are used to generate class probabilities and decision function values from the original validation set. These form the new validation set for the meta-classifier. This is then put into the meta-classifier to make predictions, which are compared against the true labels.

3.4 Prediction Phase

Because *review_text_test_doc2vec50.csv* does not have the additional 3 features mentioned, they are first added to the test set. To train each non-stacking classifier, the (full) train set is used, without splitting to get a validation set. This allows a larger training set to be used, so that better predictions can be made on the test set. Then, the models are used to make predictions for the test set. For the stacking system, the (full) training set is split into 2 halves. 4 base classifiers are trained using the first half of the training set. Then, the models are applied to the second half of the training set to get the predicted class-probabilities and decision function values. This process is repeated, training the base classifiers with the second half of the training set and generating the predicted class probabilities and decision function values for the first half. The meta-classifier is then trained using the predicted class probabilities and decision function values as attributes. Finally, the test set is fed into the base classifiers to generate the class probabilities and decision function values, which are put to the meta-classifier to generate its final predictions.

4. Results

Table 1 shows the performance of each classifier, as measured by their accuracy. Accuracy is chosen as the evaluation metric as this is a multi-class classification problem. Consequently, metrics such as precision and recall lead to matrices with multiple numbers, making direct comparisons difficult.

Accuracy		
Classifier	**Training set**	**Validation set**
0-R	0.689	0.681
Random Forest	1	0.776
SVM	0.819	0.810
Stacking	0.826	0.817

Table 1- Accuracies of different classifiers on the training and validation sets

Of the 4 models tested, the stacking classifier performed best, followed closely by SVM. Neither of these models had a significant difference between their performance on the training set and the validation set, indicating they are not overfitted. The random forest, in contrast, did overfit and had lower validation performance compared to the stacking and SVM models. Meanwhile, they all performed significantly better than the 0-R classifier, which chooses the most frequent class.

Note that these results are from a single run, and results may vary slightly when repeated due to the randomness involved in some models.

5. Discussion

Based on the results above, the 0-R classifier achieved significantly more than 33% accuracy, indicating an imbalanced dataset. Nevertheless, with about 68% of the dataset being from the largest class, the imbalance is not severe enough to make accuracy a bad metric.

All 3 non-baseline classifiers performed significantly better than the 0-R classifier, which means that they are performing reasonably. However, their performances still vary significantly, for reasons discussed below.

5.1 Random Forest

Given that Random Forest is an ensemble classifier that implements bootstrap (which reduces model variance), it is somewhat unexpected that it overfits (indicated by the large gap between the training and validation accuracies). However, this is not surprising given the hyperparameters chosen by the grid search. With only 53 features and each tree having a max_depth of 25, this is a complex model that may capture noise, resulting in overfitting.

Nevertheless, this max_depth may have been chosen during grid search because the lower values of max_depth available are insufficient to capture the patterns in the data. Thus, despite overfitting, this choice of hyperparameters results in the best performance.

5.2. SVM

Unlike Random Forest, the SVM classifier did not overfit. This indicates that the grid search chose an appropriate value for C, the regularisation hyperparameter. The C value chosen was 1, which can be considered small. Small C values indicates a larger regularisation effect, resulting in larger margins. C determines the trade-off between maximising margin and minimising training error, so picking an optimal value simultaneously maximises margins and minimises error, preventing overfitting.

Another reason why SVM outperforms Random Forest is the feature representation. As Doc2Vec transforms each instance into vectors, this feature representation is highly suitable for SVM which fits a geometric hyperplane in the feature space. This fits the SVM algorithm's assumptions, resulting in good performance.

5.3 Stacking

The stacking classifier achieved the best result out of the classifiers tested, although it improves on SVM only slightly. Part of this likely has to do with the features matching SVM very well, making it difficult to improve significantly on it. Aside from this, Logistic Regression might not be the best choice of meta-classifier, as other meta-classifiers have not been tried and compared to it.

Furthermore, the stacking classifier may also have been misled by weaker classifiers, particularly the 2 that we did not perform hyperparameter tuning for. Furthermore, the stacking classifier learns from base classifiers that were trained on only half of the training set, which may make them weaker base classifiers and thus reduce the effectiveness of the stacking classifier.

Despite these limitations, the stacking classifier generates better predictions than any stand-alone classifier. This means that it can learn patterns between the class probabilities outputted by the base classifiers and the true class label. Furthermore, it does not overfit, indicating that the value of C (the regularisation hyperparameter) identified by the grid search effectively prevents overfitting and allows the classifier to generalise.

6. Error analysis

Again, due to randomness being involved, the exact counts presented in Tables 2-5 may not hold precisely under repetition; however, they will not deviate significantly. These numbers, and the analysis, are based on the first run.

Predicted classes on the validation set			
Count	**1**	**3**	**5**
Ground Truth	475	1315	3824
0-R	0	0	5614
Meta	343	1019	4252
RF	149	629	4836
SVM	278	1008	4328

Table 2- Class label prediction counts from different classifiers

Table 2 indicates that all classifiers tested are biased towards the 5-star class, which is the majority class. Meanwhile, they are all biased against the 3-star and 1-star classes. The stacking classifier is the least biased classifier, as the distribution of its predictions are the closest to the ground truths on the validation set. This is closely followed by SVM, while Random Forest is the most biased (non-trivial) classifier.

The confusion matrix of each classifier is shown in Table 3, Table 4, and Table 5. They indicate that if the ground truth is 5-star, every classifier is more likely to make an error by predicting it as a 3-star review rather than a 1-star review. This is unsurprising, as (compared to a 1-star review) a 3-star review is more similar to a 5-star review. Meanwhile, if the ground truth is 3-star, all classifiers are more likely to classify it as 5-star instead of 1-star. This reflects the bias towards the 5-star class and against the 1-star class by all classifiers.

For the random forest classifier, the bias is severe enough to make a prediction of 5-star more likely than a 3-star prediction, even if the ground truth is a 3-star. Similarly, if the ground truth is 1-star, the random forest classifier is more likely to predict it as a 5-star review instead of a 1-star review, indicating its high bias.

The stacking classifier and SVM perform better in this aspect, as 1-star reviews are most likely to be predicted as 1-star. For the stacking classifier, if it makes an error when the ground truth is 1-star, it is more likely to predict a 3-star, which is logical as 1-star reviews are more similar to 3-star reviews than 5-star reviews. However, for SVM, 1-star reviews are more likely to be classified as 5-star than 3-star despite their higher dissimilarities, again reflecting bias towards the 5-star class.

Meta-classifier		**Predicted**		
		1	**3**	**5**
Actual	**1**	247	117	111
	3	65	719	531
	5	31	183	3610

Table 3- Confusion matrix for stacking classifier

RF-classifier		Predicted		
		1	3	5
Actual	1	127	108	240
	3	17	466	832
	5	5	55	3764

Table 4- Confusion matrix for the Random Forest classifier

SVM-classifier		Predicted		
		1	3	5
Actual	1	204	133	138
	3	53	708	554
	5	21	167	3636

Table 5- Confusion matrix for the SVM classifier

With regards to model variance, it seems that the random forest classifier has the highest variance of the tested models, as it overfits significantly. Meanwhile, the other models have not overfitted, indicating low model variance.

7. Conclusions

Based on their accuracies on the validation set, the stacking classifier performed best, with low bias and no noticeable overfitting. The SVM follows closely with a similar accuracy, along with low bias and lack of overfitting. Meanwhile, the random forest classifier performed the worst, with the largest bias and significant overfitting. However, despite outperforming other classifiers, the stacking classifier may perform even better if a more suitable meta-classifier and better-performing base classifiers are used. The stacking classifier has room for improvement, as it outperforms SVM only slightly despite also having the insight of 3 other classifiers. Furthermore, like other classifiers, it is biased towards the 5-star class, which hampers its performance.

8. References

da Silva, N., Hruschka, E. and Hruschka, E., 2014. Tweet sentiment analysis with classifier ensembles. Decision Support Systems, 66, pp.170-179.

Jones, K., 1972. A Statistical Interpretation of Term Specificity and its Application in Retrieval. Journal of Documentation, 28(1), pp.11-21.

Le, Q. and Mikolov, T., 2014. Distributed Representations of Sentences and Documents. ICML, 32, pp.1188-1196.

Mäntylä, M., Graziotin, D. and Kuutila, M., 2018. The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. Computer Science Review, 27, pp.16-32.

Mukherjee, A., Venkataraman, V., Liu, B. and Glance, N., 2013. What Yelp fake review filter might be doing?. 7th International AAAI Conference on Weblogs and Social Media,.

Nag, A., 2019. A Text Classification Approach Using Vector Space Modelling (Doc2vec) & PCA. [online] Medium. Available at: <https://medium.com/swlh/a-text-classification-approach-using-vector-space-modelling-doc2vec-pca-74fb6fd73760> [Accessed 22 May 2020].

Pranckevičius, T. and Marcinkevičius, V., 2017. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. Baltic Journal of Modern Computing, 5(2), pp.221-232.

Rayana, S. and Akoglu, L., 2015. Collective Opinion Spam Detection: Bridging Review Networks and Metadata. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.985-994.

Zhou, V., 2019. A Simple Explanation Of The Bag-Of-Words Model. [online] Towards Data Science. Available at: <https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971> [Accessed 22 May 2020].