COMP30027 Report – Sentiment Analysis

Baichuan Yu
Kenny Yu

1. Introduction

In the last decade, social media has played an increasingly role in facilitating exchange of ideas and opinions. One of the most popular social media platforms is Twitter. Across the globe, people use Twitter to create status messages called "tweets". These tweets can encapsulate people's opinions, providing insights into the current public mood on different topics. As such, tweets can serve as an avenue for companies to gather feedback. This has led to a large amount of research in the area of text classification, more specifically: sentiment analysis (1-3).

In this project, we aim to perform sentiment analysis on a database of tweets, classifying each of them into one of three sentiment categories: "positive", "negative" or "neutral". To do so, we run supervised machine learning classifiers trained on a dataset of tweets with known sentiments. Our training dataset (4) contains 21,802 instances, including 5,428 positive sentiments, 12,659 neutrals and 3,715 negatives. The dataset has two features: ID (string of numbers) and text of tweet (string). The label is the sentiment of the tweet. Our testing set contains 6,099 instances, each with an ID and a tweet. Our goal is to extract the sentiment from each instance of the testing set. Through exploration and comparing between a range of machine learning models, we hope to find the classifier that yields the best performance.

2. Method

2.1 Preprocessing

We start by eliminating as much noise from the training data as possible. All tweets are changed to lower case. Usernames (@username), URLs and all characters that are not in the English alphabet are replaced by a whitespace character. Any letters or whitespace characters occurring more than two times in a row is replaced with two occurrences. As an example, "weeeeep" would be changed to "weep".

The tweets are tokenised into lists of words, which are then lemmatised and stopwords are removed.

2.2 Feature engineering

We use both Bag of Words (BoW) and TF-IDF to convert the textual information to numeric weightage in vector format. Both unigrams and bigrams are generated. Using bigrams, we aim to target the tweets that contain negated phrases in the likes of "not good" or "not bad." Overall, 173,182 features are generated.

2.3 Feature selection

As it stands, the large number of features is likely to result in models with long training time and potentially cause overfitting problems. To select the most relevant features, we carry out a filtering experiment on the performance of our classifiers to determine the k-best features using χ^2 and mutual information. Using χ^2, the features that are more dependent on the sentiment are selected. With mutual information, the more frequently appearing features are chosen. We test a set of parameters for k: 1000, 3000, 5000 and 10000, and find that 3000 is optimal. χ^2 was chosen as the filtering method since features more dependent on the sentiment are desired.

2.4 Classifiers

2.4.1 Baseline: Zero-R

As a baseline, we classify our instances using the most frequently appearing label. For our training set, this label is "neutral".

2.4.2 Naïve Bayes

Naive Bayes is a simple model which works well on text categorization (5). The model assumes that all features are independent of

each other. Since our feature set follows a multinomial distribution, we use Multinomial Naïve Bayes in our project, with Laplace smoothing ($\alpha = 1$).

2.4.3 Support Vector Machines (SVM)

Support Vector Machines is another popular text classification technique. It aims to find the best hyperplanes that best separate the given classes. SVMs scale well to the large amount of features in this project (6), and has been shown to significantly outperform other classifiers in text classification (7). SVMs generally has very good performance when dealing with high dimensional data. Since text classification tasks are generally linearly separatable, we expect this model to perform well.

With non-linear data such as ours, the introduction of kernels is necessary. To find the most optimal kernel for our model, we use grid search to tune the hyperparameters. We investigate the combination of three kernels: 'linear', 'rbf' and 'poly', with regularization parameter = 1.0 and max iteration = 10,000 in each case. We determine 'linear' to yield the best results.

2.4.4 Logistic Regression

Logistic regression is also used for supervised machine learning. The purpose of this model is to predict the categorical or discrete dependent variable by using a given set of independent variables (8). Logistic regression is generally easy to implement, efficient to train and produces informative results that are easy to interpret. The model makes no assumptions about distributions of classes. When the data is linearly separatable, it is highly efficient even when the dataset is very large.

Again, we use grid search to tune our hyperparameters. We trial solvers: 'lbfgs', 'saga', 'sag' and 'newton-cg', penalties: 'l2' and 'none', as well as multiclass: 'multinomial' and 'ovr'. A max iteration of 10,000 is set to each combination. We determine the best model to be using 'lbfgs' for solver, 'none' for penalty and 'ovr' for multiclass.

2.4.5 Stacking

The use of stacking aims to smooth errors over a range of algorithms with different biases, combining heterogeneous classifiers with varying performance. For level 0 base classifiers, we choose the models used beforehand: Naïve Bayes, SVM and logistic regression. Logistic regression is used as the final estimator.

2.5 Model evaluation techniques

2.5.1 Training accuracy

The accuracy of the models when performed on the training set is calculated. An unsatisfactory training accuracy would suggest that the model lacks complexity and is likely to result in underfitting. On the other hand, a very high training accuracy likely indicates an overfitted model that is too complex.

2.5.2 Cross-validation

For a m-fold cross-validation, the data is split into m partitions and iteratively, one partition is held out as a test set and the other m-1 partitions are used as training data.

We use 5-fold cross-validation to evaluate each of the models based on accuracy, precision, recall and f1 score. We find that the number of instances in the testing set is approximately 28% that of the training set. Therefore, a 5-fold was chosen to best capture the relative proportions. Precision identifies the proportion of positive identifications that is actually correct, while recall measures the proportion of actual positives that is identified correctly. The F1 score combines precision and recall.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Recall = \frac{2PR}{P + R}$$

In our project, we calculate precision, recall and F1 score by both macro and weighted averaging. Due to the unbalanced nature of the training set, we expect weighted averaging to be more

reliable in encapsulating the performance of our models.

2.5.3 Confusion matrix

A confusion matrix delineates the number of correct and incorrect predictions for each class. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class. We plot a confusion matrix for each of the models evaluated on the training set. The formatting of the confusion matrix is as follows:

		Predicted		
		Positive	Neutral	Negative
Actual	Positive			
	Neutral			
	Negative			

2.5.4 Receiver operating characteristic (ROC) curve and training learning curve

The ROC curve is a graph depicting the performance of a classification model at various thresholds. The curve plots the true positive rate against the false positive rate. The Area Under the Curve (AUC) measures the ability of a classifier to distinguish between classes.

In a training learning curve, different training set sizes are plotted against the training accuracy and cross-validation accuracy. The curve shows how well the model is learning.

In this project, a ROC curve is generated for the SVM and logistic regression models, while a learning curve is plotted for all models.

2.5.5 Time complexity

The time to fit each model is calculated.

3. Results

3.1 Evaluation results

The training accuracy, as well as CV average accuracy, precision, recall and F1-score, are evaluated for each model (Table 1). Average accuracies, precisions, recalls and F1 scores are calculated by both macro and weighted averaging. In addition, the time complexities to fit each model were also calculated.

We observe that in almost all cases, models using BoW outperforms their TF-IDF counterpart. Excluding stacking, logistic regression appears to perform the best, having the highest values in all evaluation metrics. By examining the confusion matrices, it seems that an overwhelming number of instances are being predicted as “neutral”, resulting in a plethora of false negatives.

		Zero-R	Multinomial Naïve Bayes	SVM	Logistic regression	Stacking
Training accuracy	BOW	0.5806	0.6998	0.7777	0.7957	0.7860
	TFIDF	0.5806	0.6142	0.6637	0.7897	0.7826
CV accuracy	BOW	0.5806	0.6730	0.7048	0.7120	0.7194
	TFIDF	0.5806	0.6006	0.6414	0.7222	0.7283
CV precision (macro)	BOW	0.1935	0.6365	0.6978	0.7020	0.7179
	TFIDF	0.1935	0.7780	0.6912	0.7210	0.7376
CV precision (weighted)	BOW	0.3372	0.6783	0.7018	0.7111	0.7172
	TFIDF	0.3371	0.7041	0.6689	0.7245	0.7322
CV recall (macro)	BOW	0.3333	0.6457	0.5975	0.6242	0.6308
	TFIDF	0.3333	0.3650	0.4398	0.6252	0.6376
CV recall (weighted)	BOW	0.5801	0.6748	0.7012	0.7139	0.7216
	TFIDF	0.5806	0.5998	0.6420	0.7200	0.7287
CV F1-score (macro)	BOW	0.2448	0.6382	0.6229	0.6465	0.6642
	TFIDF	0.2448	0.3108	0.4368	0.6564	0.6730
CV F1-score (weighted)	BOW	0.4266	0.6770	0.6849	0.7015	0.7117
	TFIDF	0.4265	0.4747	0.5667	0.7082	0.7155
Confusion matrix	BOW	0 5428 0 0 12659 0 0 3715 0	3463 1804 161 1771 9331 1557 214 1037 2464	3223 2149 56 868 11479 312 87 1374 2254	3288 2106 34 688 11655 316 65 1246 2404	3326 2078 24 836 11429 394 69 1264 2382
	TFIDF	0 5428 0 0 12659 0 0 3715 0	641 4787 0 72 12583 4 6 3541 168	1837 3573 18 430 12126 103 54 3154 507	3123 2271 34 642 11718 299 62 1278 2375	3182 2224 22 740 11531 388 49 1317 2349
Time complexity	BOW		0.024	17.39	9.595	126.359

Table 1. Evaluation results for each model. The table contains the training accuracy, CV average accuracy, precision, recall and F1 score, as well as time complexity for each model used.

3.2 Learning curves

Learning curves for Naïve Bayes, SVM, logistic regression and stacking are generated (Figure 1). The curves all converge to a reasonable accuracy, suggesting that there is a good trade-off between bias and variance in all models.

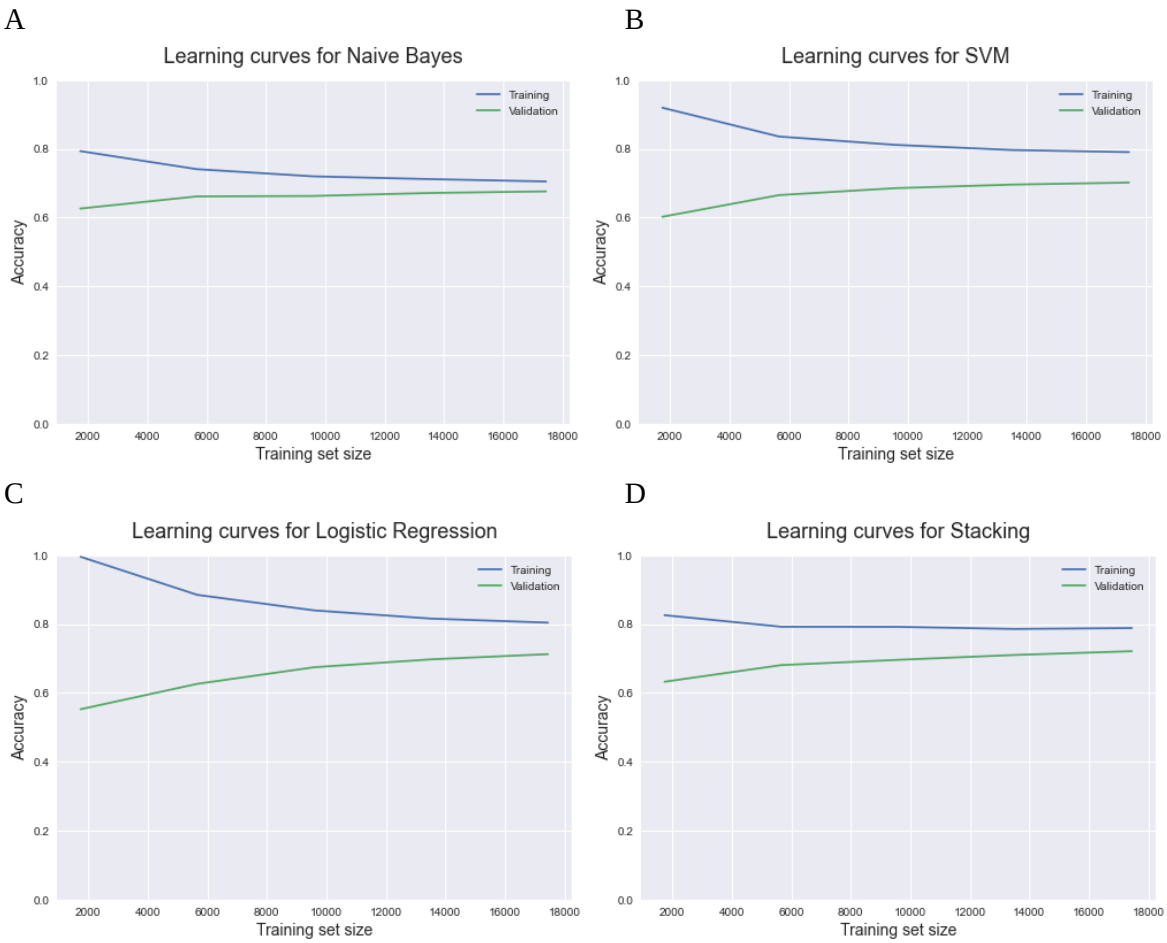
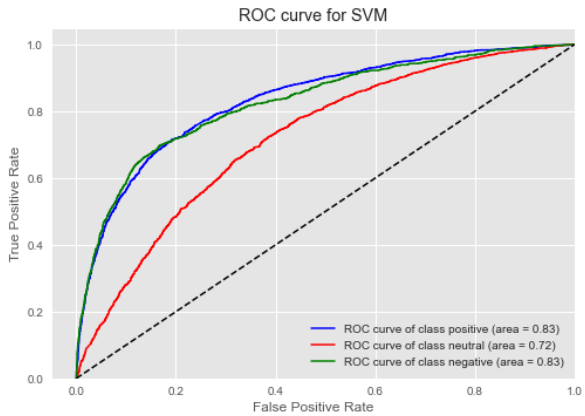


Figure 1. Learning curves. A) depicts the learning curves for Naïve Bayes, B) for SVM, C) for logistic regression, and D) for stacking.

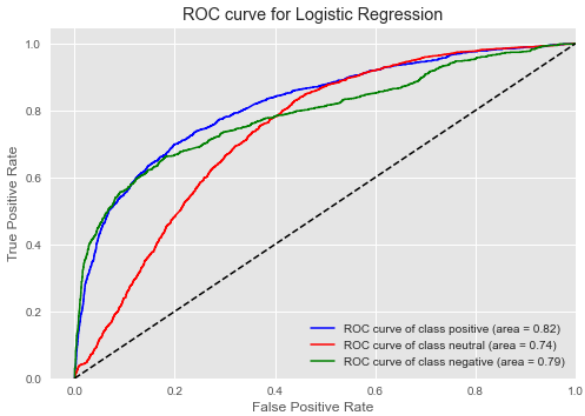
3.3 ROC curves

ROC curves are generated for SVM, logistic regression and stacking (Figure 2). It seems that the performance of SVM and logistic regression in generating true positive and false positive is rather similar. Stacking, however, outperforms both.

A



B



C

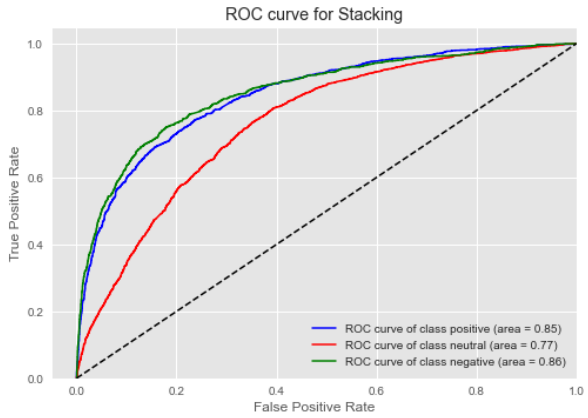


Figure 2. ROC curves. A) depicts the ROC curve for SVM, B) for logistic regression, and C) for stacking.

4. Evaluation

The learning curves generated for the models all converge to a satisfactory accuracy, delineating a good bias-variance trade-off. From observing model performance, we observe that logistic regression performs better than SVM, while Naïve Bayes performed the worst. As expected, stacking achieved the best performance. One reason that Naïve Bayes did not perform as well as the others may be due to to the assumption of conditional independence of features. This assumption may lead to a model that lacks the necessary complexity to capture the patterns of the dataset. Contrary to other studies (9, 10), SVM did not perform the best. This may be due to the fact that SVM does not handle large data sets as well as logistic regression. Moreover, noise tends to have more impact on the performance of SVM. Inspecting the ROC curves, we learn that while SVM and logistic regression have similar performance in generating true positives and false positives, stacking offers a better performance when compared to both. We also note that in all cases, models using BoW outperforms those using TD-IDF.

For all models, the confusion matrices suggest that there is a general trend to classify instances as being "neutral". We observe from Figure 3 that the distribution of sentiments in our training dataset is highly unbalanced. Namely, there are vastly more "neutrals" than the other sentiments.

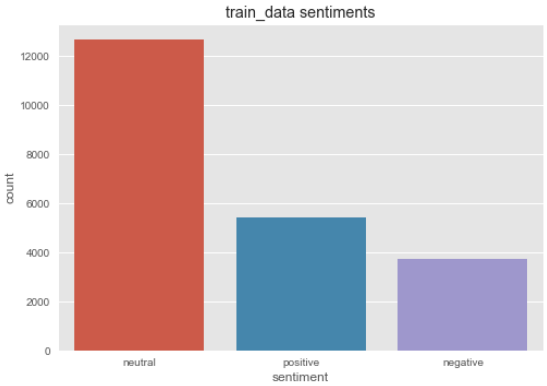


Figure 3. Distribution of sentiments in training set

We postulate that this unbalance in distribution may have resulted in leniency or bias towards the sentiment "neutral", causing underfitting. We test this hypothesis by balancing the number of "neutral", "positive" and "negative" tweets in our training set. Indeed, there is a boost in performance, especially in recall calculated by macro-averaging. The slight decrease in average accuracy is potentially another indication of model bias if the full unbalanced training set was used. We include the evaluation results of the stacking classifier using BoW in Table 2. The improvement is due to the reduction of false negatives, namely that less instances are being wrongly classified as "negative".

In the context of sentiment analysis, we should be more interested in tweets that have "negative" sentiments since they are more likely to contain violent, provocative, explicit or sensitive content. Thus, false negatives: classifying tweets with "negative" sentiment as "neutral" or "positive", should be regarded as the more important error.

As a whole, the results obtained in our project should be reliable. The evaluation metrics for all models are similar and lie within an expected range. The training accuracies are not too high, which would point to overfitting, and not too low, which would indicate underfitting and a model that lacks complexity.

In our project, models using BoW consistently performed better than those using TF-IDF. A potential reason may be that the hierarchy of word importance as determined by TF-IDF is rendered unimportant by the use of feature selection through χ^2. For future research, however, it would be beneficial to delve deeper into the reasons behind this decrease in performance, especially since it is the default word vectorisation method chosen by many groups (11-14).

	Stacking (Using BoW and balanced training set)
Training accuracy	0.8074
CV accuracy	0.6818
CV precision (macro)	0.6790
CV precision (weighted)	0.6837
CV recall (macro)	0.6872
CV recall (weighted)	0.6827
CV F1-score (macro)	0.6809
CV F1-score (weighted)	0.6855
Confusion matrix	3107 488 121 542 2668 506 159 331 3225

Table 2. Performance of stacking when using a balanced BoW training set.

5. Conclusion

This study aimed to explore the performance of different machine learning models in extracting tweet sentiment. Ultimately, we find that out of Naïve Bayes, SVM and logistic regression, the latter offers the best performance on the dataset we obtained. Better still was the performance yielded by using stacking. We find that by using a more balanced training set, we can reduce model bias, leading to less instances of false negatives.

6. References

1. Habib A. Performance Analysis of Machine Learning Algorithms for Movie Review. International Journal of Computer Applications. 2020;177:7-10.
2. Prakash TN, Aloysius A. Applications, Approaches, and Challenges in Sentiment Analysis (AACSA). International Research Journal of Modernization in Engineering Technology and Science, ISSN.2582-5208.
3. Sahu TP, Ahuja S, editors. Sentiment analysis of movie reviews: A study on feature selection & classification algorithms. 2016 International Conference on Microelectronics, Computing and Communications (MicroCom); 2016: IEEE.
4. Rosenthal S, Farra N, Nakov P. SemEval-2017 task 4: Sentiment analysis in Twitter. arXiv preprint arXiv:191200741. 2019.
5. Manning CD, Schutze H. Foundations of Statistical Natural Language Processing. 2002:91-2.
6. Joachims T, editor Text categorization with Support Vector Machines: Learning with many relevant features. Machine Learning: ECML-98; 1998 1998//; Berlin, Heidelberg: Springer Berlin Heidelberg.
7. Yang Y, Liu X. A re-examination of text categorization methods. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval; Berkeley, California, USA: Association for Computing Machinery; 1999. p. 42–9.
8. Hossen MS, Chowdhury MNA, Sristy AM, Jahan N. Sentiment Analysis using Machine Learning and NLP for Digital Education. 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Computing Methodologies and Communication (ICCMC), 2022 6th International Conference on: IEEE; 2022. p. 902-8.
9. Jaya Hidayat TH, Ruldeviyani Y, Aditama AR, Madya GR, Nugraha AW, Adisaputra MW. Sentiment analysis of twitter data related to Rinca Island development using Doc2Vec and SVM and logistic regression as classifier. Procedia Computer Science. 2022;197:660-7.
10. Sudhir P, Suresh VD. Comparative study of various approaches, applications and classifiers for sentiment analysis. Global Transitions Proceedings. 2021;2(2):205-11.
11. Li G, Liu F. Application of a clustering method on sentiment analysis. Journal of Information Science. 2012;38(2):127-39.
12. Liu S, Lee I, editors. Email Sentiment Analysis Through k-Means Labeling and Support Vector Machine Classification. Cybernetics and systems; 2018/01/01/2018; Great Britain: Taylor & Francis.
13. Nigam K, McCallum AK, Thrun S, Mitchell T. Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning. 2000;39(2):103-34.
14. Whitelaw C, Garg N, Argamon S. Using appraisal groups for sentiment analysis2005.