Exploring Support Vector Machine for Binary Sentiment Classification

Ahnaf Mozib Samin, Ayça Avcı, Shantanu Nath
(s4996763), (s4505972), (s4998405)
a.m.samin@student.rug.nl, a.avci@student.rug.nl, s.n.nath@student.rug.nl

Abstract

With the growing popularity of social networking websites, sentiment analysis is now being broadly applied to solve many problems in businesses, government, and research and thus has drawn the attention of the research community. However, it is still considered a challenging task due to the unclear meaning of texts and ambiguity in detecting the suitable polarity of sentiments. This paper explores Support Vector Machine (SVM), a supervised machine learning algorithm, in-depth to find out the best performing model with changing several parameters using a publicly available text corpus in English collected from Twitter. We achieve 82.3% accuracy and 82% F1-score using Support Vector Classification (SVC) implementation in scikit-learn with linear kernel and Term Frequency-Inverse Document Frequency (TF-IDF) feature. In addition, we also emphasize experimenting with several input features that help us further improve the accuracy. We obtain 83.3% accuracy and 83% F1-score using TF-IDF with N-gram features and SVC algorithm. These results demonstrate the competitiveness of the SVM algorithm for sentiment analysis.

1 Introduction

Sentiment analysis utilizes the methods in Natural Language Processing (NLP), text analysis, and Computational Linguistics (CL) for automatically classifying the sentiments into either binary or multiple classes. This technique has been proven to be very useful in business since it can provide insights into public opinions. The reviews extracted from different websites are used to monitor the current trends, likes, and dislikes which can be used for improvement in the quality of products/services. Nowadays marketing approaches have also been shaped by public opinions. The benefit of sentiment analysis is not only confined in business but also has widely been used in different sectors including the Government agencies for decision making.

For developing a sentiment analysis system, it is imperative to have a large-scale balanced text corpus as well as a well-developed model to get good accuracy. Nowadays, most researchers are utilizing large corpora and training deep learning models with them. P. Chen et al. develop the Recurrent Attention on Memory (RAM) model combining multiple attentions with a Recurrent Neural Network and achieve state-of-the-art accuracy on four datasets, namely SemEval 2014 (two datasets), Twitter dataset, and Chinese news comment dataset (Chen et al., 2017). C. Colón-Ruiz and I. Segura-Bedmar compare different deep learning approaches including BERT, CNN, and LSTM to classify drug reviews in 2020 (Colón-Ruiz and Segura-Bedmar, 2020). Although deep learning has become the state-of-the-art approach with a large amount of data, SVM is still one the of most widely used supervised machine learning techniques for text classification for languages that are lacking large corpora (Ahmad et al., 2018). As only a few languages have considerably large corpora for sentiment analysis, it is worthwhile to experiment with the SVM-based model trained on a rather smaller corpus.

The contribution of this paper is two-fold. First, we implement an SVM-based sentiment analysis

Table 1: Data set divided into ”positive” and ”negative” classes for binary sentiment analysis

Classification Type	Classes	No. of Samples							
		Train		CV		Test		Total Samples (per class)	
		Freq.	(%)	Freq	(%)	Freq	(%)	Freq.	(%)
Binary	Positive	1881	49.0	470	48.9	617	51.4	2968	49.4
	Negative	1959	51.0	490	51.1	583	48.6	3032	50.6

model on an English text corpus with 6000 samples and then analyze its performance. We compare the three different types of SVMs namely Support Vector Classification (SVC), Linear Support Vector Classification (Linear SVC), and Support Vector Regression (SVR). SVC is found to be our best-performing model with 82.3% accuracy. The impacts of the ”C” in SVM and different kernels on accuracy have also been found out using SVC on our dataset.

Secondly, we explore different input features including TF-IDF, bag-of-words, n-gram, pos-tagging, and several combinations of these approaches. In our experiment, we find that SVM with TF-IDF and 3-gram features yields the best accuracy on our test, getting 83.3% accuracy and 83% F1-score.

In Section 2, information about data set used in the model training and testing is provided. In Section 3, we explain experimental setup and evaluation methods that are used to assess model performance. In Section 4, we provide our results, and in Section 5, we discuss our findings. In Section 6, we make conclusions on the results that are presented in Section 4.

2 Data Set

For the binary sentiment classification task, we utilize a publicly available data set containing 6000 textual data (in English) collected from a popular social networking platform - “Twitter”. These text samples have already been annotated. We analyze the sentiments (either positive or negative) of the samples. We preprocess the dataset by filtering out the stop words, punctuations, lower-casing all the letters, applying lemmatization for our task. All the samples in the data set are randomly split into three sets namely train set, cross-validation (CV) set, and test set. 20% of the whole data set is kept for the test set. We perform 5-fold cross-validation with our train and CV sets. To keep the fairness of the experiment, we only use our test set for the final evaluation. Table 1 reports the distribution of the samples into these three sets for ”positive” and ”negative” classes. As we can see from the table, we ensure an almost even distribution of samples for each class, thus, prepare a balanced data set for our experiments.

3 Method

To run the experiment, we split our data set into a training set (64%), CV set (16%), and test set (20%). We use the CV set for parameter tuning and do not use the test set till we decide on the best parameters for our model. The main purpose of not including the test set in parameter tuning is to get an unbiased estimate of model performance with the best parameters from the test set. We experiment using different feature vectors for the classification task. Two of the feature vectors are generated by the following tools provided by scikit-learn library in Python:

1. **Bag of Words (Count) Vectorizer:** Converts text data into a vector of token counts, the number of times a word appears in the document.

2. **TF-IDF (Term Frequency - Inverse Document Frequency) Vectorizer:** Converts text into a vector of overall document weightage of a word, weights the word counts by a measure of how often they appear in the document.

We also use custom features that are generated by us, including the count of sentences, count

Table 2: Comparison between the majority baseline and different SVM approaches such as Support Vector Classification (SVC), Linear Support Vector Classification (Linear SVC) and Support Vector Regression (SVR), with TF-IDF input features. Here, F1 denotes the F1-score of our models. Results on both CV and test sets are reported.

	CV		Test	
Model	F1	Accuracy (%)	F1	Accuracy (%)
Majority Baseline	-	51.1	-	51.4
SVC	0.81	81.0	0.82	82.3
Linear SVC	0.82	82.1	0.81	81.0
SVR	0.79	79.3	0.79	79.2

of words in a sentence, entity counts (counts of company, person, etc.), and count of the adjectives for each text data. We combine them with the other feature vectors to experiment with whether the accuracy changes significantly. After model training, we test our data with a test set, and print accuracy, precision, recall, and F1-score values of the model. We compare these results in Section 5. We use cross-validation with CV set to test the trained model's ability to predict new data which is not used in estimation, to identify problems like selection bias, over-fitting, and to provide an insight on how the trained model will generalize on an independent data set.

We train a Support Vector Machine (SVM) classifier to perform the binary classification task. SVM algorithm relies on creating a line or a hyperplane which separates the data into two classes. We experiment using Support Vector Classifier (SVC) from scikit-learn SVM package. We use different kernels namely "linear" and "Radial Basis Function" (rbf), and different C and gamma values as a parameter to observe the change in the accuracy. A kernel is a function that takes low dimensional input space, transforms it to a higher dimensional space. It converts not separable problem to a separable problem. It is especially used for non-linear separation problems. C value controls the trade-off between smooth decision boundary and classifying training points correctly. A large value of C means we will get more training points correctly. Gamma parameter is only used when the kernel is "rbf". It defines how far the influence of a single training example reaches. If the gamma value is high, it means that every point has a close reach. If it is too high, and decision boundary will be dependent on the points that are very close to the hyperplane. This will result in ignoring the points that are very far from the decision boundary since closer points get more weight. Conversely, if the gamma value is low, every point has a far reach, thus far away points get effective weight, decision boundary will be more linear.

We also used the LinearSVC package and Support Vector Regressor (SVR) packages from scikit-learn to see how these models perform on our data set. Both LinearSVC and SVC are supposed to optimize the same problem, but in fact, all LinearSVC estimators penalize the intercept, whereas SVC ones do not. This leads to a different mathematical optimization problem, and thus different results. LinearSVC tends to be faster to converge the larger the number of samples.

4 Results

In this section, we are going to show the outcomes of our experiments. In Table 2, the F1-score and accuracy are presented calculated using the majority baseline and different types of SVM algorithms namely Support Vector Classification (SVC), Linear Support Vector Classification (Linear SVC), and Support Vector Regression (SVR). TF-IDF input feature has been used in all cases. We experiment with different parameters of these SVM implementations and report only the best results on Table 2 Here, our majority baseline is the model which always predicts the most frequent class. As seen from the table, SVC outperforms the majority baseline by a large margin getting 81% of both accuracy and f1-score

Table 3: Precision (P), recall (R) and F1-score (F1) calculated per class on our test set for the model trained with SVC algorithm with TF-IDF features.

Classes	P	R	F1
Positive	0.83	0.81	0.82
Negative	0.82	0.84	0.83

Table 4: Average F1-scores (F1) and accuracy calculated for binary classification. Here, model trained with SVC algorithm (where kernel="linear") with multiple features.

	CV		Test	
Input Feature	F1	Accuracy (%)	F1	Accuracy (%)
TF-IDF	0.80	79.6	0.82	82.3
TF-IDF + 3-gram	0.80	80.3	0.83	83.3
TF-IDF + pos	0.80	80.1	0.83	83.2
TF-IDF + 3-gram + pos	0.79	79.4	0.83	82.3
Bag-of-words	0.76	76.4	0.79	78.9
Bag-of-words + pos	0.79	79.1	0.76	80.7
Bag-of-words + 3-gram + pos	0.80	80.7	0.79	79.1
Bag-of-words (diff. cutoff per feature) + pos	0.76	76.5	0.79	79.3
Custom features + Bag-of-words	0.78	78.3	0.79	79.3

whereas the latter only achieves 51.4% accuracy on the CV set. For the SVC, we use the default settings (kernel is linear and C=1.0). We try out different values of C from 0.5 to 50 and find no impact of it on accuracy using our CV set. The Linear SVC-based model yields the highest accuracy and F1-score, getting 82.1% accuracy and 82% F1-score , whereas the SVR results in 79.3% accuracy and 79% F1-score. However, on our test set, the SVC model performs better, achieving 82.3% of accuracy and 82% F1-score while the Linear SVC approach gets 81% of accuracy and f1-score. Again, our SVR model performs poorly amongst the three models, getting 79.2% accuracy.

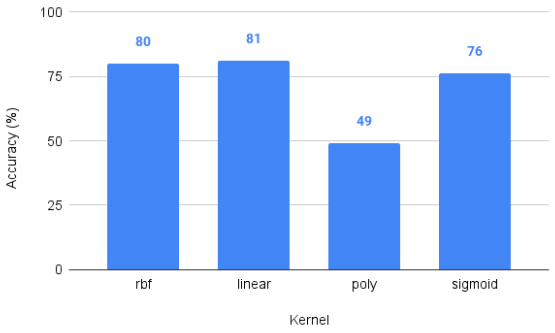

Figure 1: Impact of four types of kernels on accuracy using SVC model and TF-IDF features on the CV set

We can see the effect of four types of kernels on accuracy using the SVC implementation and TF-IDF features on the CV set. The four kernels rbf, linear, poly and sigmoid achieve 80%, 81%, 49%, and 76% accuracy, respectively. So, the linear kernel gets the best accuracy.

Since we now have our best-performing model to be SVC on our test set, we calculate the precision, recall and F1-score per class using it with TF-IDF features on the test set (See Table 3). While performing the text classification, we see 83% of precision and 81% of recall for the "positive" class. Similarly, the "negative" class scores 82% of precision but 84% of recall. The negative class is predicted more correctly than the positive class by our model, getting an F1-score of 83% for the negative and 82% for the positive class.

From Table 4, we can see that the F1-score and

accuracy for different types of input features on both CV and test sets. Overall, TF-IDF achieves better accuracy than bag-of-words. More specifically, TF-IDF with 3-gram gets the highest accuracy of 83.3% and an F1-score of 83% on our test set. By using only the bag-of-words feature, we get 78.9% accuracy on the test set which is the lowest. The same trend with TF-IDF being better than the bag-of-words has been seen on our CV set as well. Using the custom features, we get an accuracy of 79.3% and an F1-score of 79%.

5 Discussion

We experiment with the SVM algorithm with various input features using an open-sourced and well-balanced data set containing 6000 annotated textual data. From our experiment, we can observe that Support Vector Classification (SVC) performs better than the other two implementations of SVM (See Table 2). SVM has three implementations with different functionalities in the scikit-learn package provided by Python. SVC and Linear SVC are used for classification whereas Support Vector Regression (SVR) is used for regression problems. But, we can also use SVR by converting the classes into numbers like '0' and '1', negative and positive respectively. We exploit both TF-IDF and bag-of-words features and find that TF-IDF produces better accuracy for binary sentiment classification. However, using only such a small-sized data set, we cannot come to a conclusion on which feature vectors are more suitable for the SVM to yield better performance.

We also analyze the different parameters of the SVC to see whether our model shows the best result. We cannot observe any difference for the different values of C. We argue that as the dataset is small, it becomes easier for our classifier to make a linear separation and so we see no impact of this parameter. Another parameter gamma has also been found to have minimal effect on accuracy. While experimenting with kernels, we find that the Linear kernel shows the best result compared to the other ones.

Apart from these, we also put our attention on feature analysis. we preprocess the sentence by removing stopwords, punctuation, and applying lemmatization. Therefore, it eliminates those words which are not important to learn. Also, the Parts of Speech (POS) tagger provides more information about the data. Suppose, "Drinking is injurious. Therefore, I don't drink." is a sample which we want to classify into its sentiment. Here, "drinking" is a noun, and "drink" is a verb. Though these words contain the same lemma, their functionalities are different. We apply Name Entity count which provides some extra features that help understand the difference between two similar words. Another example, "Apple has released iPhone 13 when I am drinking Apple Juice." Here, the first apple is a name of an organization where the second apple is the name of a fruit. So, the first Apple can be distinguished if we apply POS tagger. Similarly, the number of adjectives provides extra features from a text.

Though Adding custom features gives us more information about the data, accuracy is dropped in our experiment. Another findings is that it takes a huge amount of time when we use TF-IDF vectorizer rather than Bag-of-words. Using the "svm.coef_" attribute with the linear kernel, we can visualize the most contributing features in both directions (positive and negative) and weights. We see the most important features as expected.

Also, our best model does not provide always accurate results. For example, a review with id 101.txt from reviews.txt file is a negative review. But our model predicts it as Positive review. On the other hand, 1838.txt is a positive review predicted as negative review.

For both "positive" and "negative" classes, we do not observe a specific class performing significantly different than the other one (See Table 3. This implies the fact that the data set we have used in our experiment is well-balanced.

6 Conclusion

In this paper, we implement the SVM algorithm for sentiment analysis using an open-source English text corpus with 6000 samples. We also investigate the performance of several types of SVM implementations in scikit-learn such as Support Vector Classification (SVC), Linear Support Vector Classification (Linear SVC), and Support Vector Regression. This study suggests that SVC per-

forms the best amongst these algorithms, with an accuracy of 82.3% and an F1-score of 82% on our test set. The default settings with linear kernel and the value of C as 1.0 are used to get the best result. Moreover, three types of kernels are studied, and it is found that linear kernel obtains higher accuracy. In addition, we experiment with the parameter "C" in SVM and observe no impact on accuracy using our corpus. Apart from the experimentation with SVM, we also evaluate varied types of input features including TF-IDF, bag-of-words, n-gram, pos-tagging, custom features, and several combinations of these approaches. We find that TF-IDF with n-gram features improves the accuracy of our SVC model to 83.3% and F1-score to 83%. In the future, the performance of a neural network on our small corpus can be studied to better understand the performance of SVM in comparison with NN.

References

Munir Ahmad, Shabib Aftab, Muhammad Salman Bashir, and Noureen Hameed. 2018. Sentiment analysis using svm: A systematic literature review. *International Journal of Advanced Computer Science and Applications*, 9(2).

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 452–461.

Cristóbal Colón-Ruiz and Isabel Segura-Bedmar. 2020. Comparing deep learning architectures for sentiment analysis on drug reviews. *Journal of Biomedical Informatics*, 110:103539.