Tweets Sentiment Classification

Anonymous

1 Introduction

Today, various machine learning techniques have been invented. These techniques have been applied in many industries, such as topical categorization, scam email detection, movie review analysis, etc. Through these algorithms, we can develop and utilize data more efficiently.

The efficiency of different algorithms in different scenarios is different, and the requirements for supervision levels are also different. For example, Naive Bayes and KNN are supervised, K-Means Clustering is unsupervised, and the Self-Training approach is semi-supervised. Different sample sizes also affect the performance of the classifier. For example, when the sample size is small, KNN may perform better than Naive Bayes, but as the sample size increases, Naive Bayes will outperform KNN (Kalaivani & Shunmuganathan, 2013).

In this paper, we use data consisting of 144,000 tweets (Blodgett et al., 2016), of which 100,000 unlabeled samples; 40,000 labeled training samples; and 4,000 labeled test samples. Since the experimental dataset includes three different feature representations, we use two of them here. The first is the TFIDF dataset, where each tweet is converted into a 1000-dimensional vector representation, each dimension uses a TFIDF value to represent the frequency of a word. The second one is the Embedding dataset, which uses a pre-trained language model to map the raw tweets to a 384-dimensional vector. Such a mapping can enable similar tweets to be close in a 384-dimensional space.

We use different machine learning algorithms for sentiment classification of tweets in this paper, such as determining whether the content of a tweet is positive or negative, and find the strengths and weaknesses of different machine learning paradigms and trying to see if we can effectively combine labeled and unlabeled training data, so that can improve the Twitter sentiment classification.

2 Literature Review

The data set used in this paper are from the resource published by Blodgett et al. (2016).

In recent years, sentiment classification has become a relatively popular research area.

Pang et al. (2002) used a supervised learning approach to categorize film reviews for positive and negative sentiment. Among the methods, Naive Bayes performed the worst and SVMs performed the best.

Rothfels and Tibshirani et al. (2010) used an unsupervised learning method to classify Chinese product reviews and English film reviews respectively, the results were not very optimistic, and the iterative reclassification algorithm failed to improve the classifications.

Jiang and Yu et al. (2011) proposed a Target-dependent tweet sentiment classification method. In previous studies, most experiments were based on the target-independent assumption, but in reality, tweets are not independent, it is often associated with images, tags, and previous tweets posted by the user. Combining these data, the target-dependent strategy performs better than the target-independent strategy.

3 Methods

We experimented with five different machine learning algorithms, including one baseline model; two supervised learning algorithms; one unsupervised learning algorithm; and one semi-supervised learning algorithm. According to the results of previous studies, each algorithm is proven to be effective and feasible for text classification tasks.

We use four evaluation metrics here, accuracy, precision, recall, and F1 score.

3.1 Baseline Model

The baseline model we used here is the majority class baseline method, also known as the 0-R strategy, which classifies all instances according to the most frequent class label in the observed data.

3.2 K-Nearest Neighbors

The K-Nearest Neighbor (KNN) is a traditional machine learning algorithm that selects the nearest k neighbors by calculating the distance between the new instance and the

surrounding neighbors, then using the majority vote method to find which is the most frequent class label among the neighbors, and assign this class label to the new instance. We use this algorithm here for comparison with the Naive Bayes algorithm.

3.3 Gaussian Naïve Bayes

Gaussian Naïve Bayes (GNB) is a probabilistic classifier that based on the Bayes' rules. The method assumes that the features are independent of each other, and we learn the model parameters θ through the given training set so that we can maximize the likelihood of the data, and then predicts the input X based on the learned model.

Let us assume $Y = \{y_1, y_2, \dots, y_M\}$ is the class labels, and $X = \{x_1, x_2, \dots, x_k\}$ is the observation instance, and x_k is the features of X. Then the underlying probabilistic model is:

$$P(X,Y) = \prod_{m=1}^{M} P(y^m) \prod_{k=1}^{K} P(x_k^m | y^m)$$

In Gaussian Naïve Bayes, we assume that the y_m is drawn from the Bernoulli distribution, and x_k is drawn from the Gaussian distribution, then the formula is:

$$P(X,Y) = \phi^y (1-\phi)^{1-y} \prod_{k=1}^{K} \frac{1}{\sqrt{2\pi\sigma_{k,y}^2}} e^{-\frac{(x_k-\mu_{k,y})^2}{2\sigma_{k,y}^2}}$$

Where ϕ is the frequencies of classes observed in the training data; μ is the mean feature value of x_k under each class y; and σ is the standard deviation.

To deal with unseen features, we use the epsilon smoothing strategy, that is when we calculate $P(x_k | y) = 0$, we will use a very small number to replace the 0 value, here we use $\epsilon = 1e^{-9}$.

The Naive Bayes algorithm has been proven that perform well in text classification (Lewis, 1998), so we will implement this algorithm in our paper.

3.4 K-Means Clustering

As a contrast, we implement an unsupervised algorithm here.

K-means clustering is an unsupervised algorithm, that group similar data points together. The algorithm implement steps are as follow:

(1) Choose k points as the cluster initial centroids;
(2) Assign each instance to the closet cluster based on the distance with the centroid;
(3) Recalculate the centroid of each cluster;
(4) Repeat step (2) and (3) until the centroids becomes stable.

The centroid typically means the average value of the points in that cluster, and usually randomly selected at the beginning.

3.5 Self-Training

Semi-supervised learning is a strategy that attempts to combine labeled and unlabeled data to train algorithms. Self-training is one such algorithm, and is run as follows:

(1) Use the labeled to train the classifier;
(2) Use the trained classifier to classify the unlabeled data, and select the samples that exceed the threshold;
(3) Take out the selected sample from the unlabeled data set and merge them into the labeled data set;
(4) Retrain the classifier with the new labeled data set;
(5) Repeat steps (2) to (4) until convergence.

The advantage of self-training is to act as a wrapper, and this is why we chose this algorithm. In this paper, we employ GNB and KNN as its core classification algorithm.

4 Results

In this section we will summarize the resulting data obtained from the experiments, with in-depth analysis and discussion in section 5.

According to our experimental results, supervised learning algorithms perform the best, followed by semi-supervised learning algorithms, and unsupervised learning algorithms perform the worst. This result holds for both Embedding and TFIDF feature representations (Figure 1).

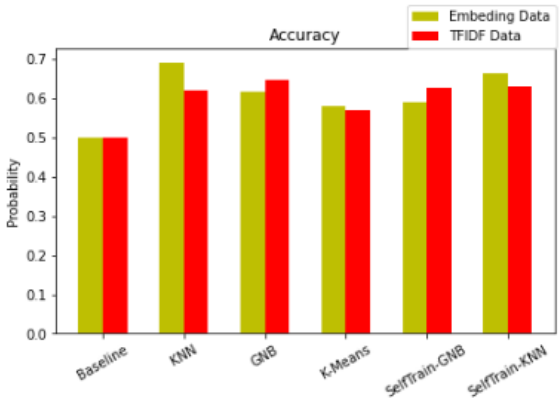

Figure 1 - Accuracy of different machine learning algorithms. Evaluate with hold out strategy.

In our experiments, we find that with the exception of GNB, which performed well on the TFIDF dataset, all other algorithms performed better on the embedding dataset.

We also find that using unlabeled data does not help to improve the performance of the classifier and even reduced the accuracy of the classifier. The performance of the Self-Training model based on GNB decreases as the number of unlabeled samples increases (Figure 2). The performance of the Self-Training model based on KNN increases slowly when the number of unlabeled samples is less than 50,000, after exceeding 50,000, it begins to fluctuate greatly and shows a decreasing trend (Figure 3).

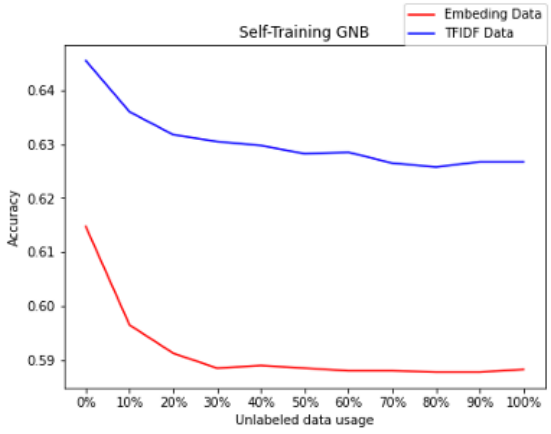

Figure 2 - Self-Training based on GNB with different amount unlabeled data sample usage (The X-axis represents the proportion of the unlabeled dataset used, e.g., 50% means $10^5 \times 50\%$, where 10^5 is the size of the unlabeled dataset).

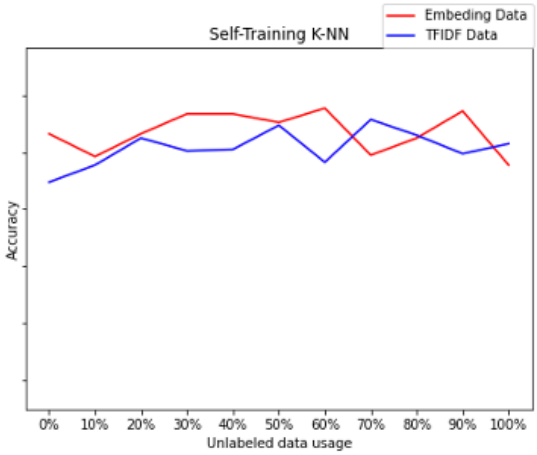

Figure 3 - Self-Training based on KNN with different amount unlabeled data sample usage (The X-axis represents the proportion of the unlabeled dataset used, e.g., 50% means $10^5 \times 50\%$, where 10^5 is the size of the unlabeled dataset).

5 Discussion

Different machine learning paradigms have different strengths and weaknesses. When the labeled data is available, the supervised learning algorithm might be a good choice. But in reality, we often encounter situations where we lack sufficient prior knowledge to manually label categories, in which case unsupervised learning algorithms will be able to help us. Or maybe we have enough prior knowledge, but we can't label all the data because the cost of manually labeling the categories is too high, then a semi-supervised learning algorithm will be able to help us. However, it should be noted that the efficiency of semi-supervised learning algorithms in some situations may not be as good as supervised learning algorithms.

According to our experimental results, unlabeled samples not help the model to improve performance. The reason for this may be that the Self-Training model learns wrong class label during the learning process. As can be seen from the resulting data (see Table 1), the model under the supervised learning paradigm is less than 70% accurate, and the classifier used by Self-Training is likely not able to accurately classify the unlabeled samples, so the newly merged training set is not valid anymore, and in the following iterations that situation happened again and again, resulting in the model accuracy decline. This inference can be verified by observing the confusion matrixes.

Embedding dataset	Supervised		Unsupervised	Semi-Supervised	
	KNN	GNB	K-Means	Self-training KNN	Self-training GNB
Accuracy	0.68	0.61	0.57	0.66	0.58
Precision	0.69	0.61	0.59	0.68	0.59
Recall	0.68	0.61	0.57	0.66	0.58
F1	0.68	0.61	0.56	0.65	0.57

Table 1 - Performance of different machine learning algorithms with Embedding dataset. Evaluate with hold out strategy.

TFIDF dataset	Supervised		Unsupervised	Semi-Supervised	
	KNN	GNB	K-Means	Self-training KNN	Self-training GNB
Accuracy	0.61	0.64	0.56	0.63	0.62
Precision	0.62	0.64	0.59	0.63	0.62
Recall	0.61	0.64	0.56	0.63	0.62
F1	0.61	0.64	0.53	0.63	0.62

Table 2 - Performance of different machine learning algorithms with TFIDF dataset. Evaluate with hold out strategy.

Comparing the confusion matrixes of GNB algorithm and Self-Training based on GNB, that before using unlabeled data, the value of $FN + FP = 939 + 602 = 1541$, and after using 100,000 unlabeled data, this value increases to 1647 (see Table 3 and 4). It is obvious that the bias of the model increases. The same results can also be seen in the KNN-based Self-Training model, $FN + FP = 491 + 757 = 1248$ before using unlabeled data, increased to 1348 after using unlabeled data (see Tables 5 and 6).

GNB	Negative	Positive
Negative	1388	602
Positive	939	1061

Table 3 - Confusion matrix of GNB model with Embedding dataset.

Self-Training GNB	Negative	Positive
Negative	1496	504
Positive	1143	857

Table 4 - Confusion matrix of Self-Training GNB model with Embedding dataset.

KNN	Negative	Positive
Negative	1243	757
Positive	491	1509

Table 5 - Confusion matrix of KNN model with Embedding dataset.

Self-Training KNN	Negative	Positive
Negative	1011	989
Positive	359	1641

Table 6 - Confusion matrix of KNN model with Embedding dataset.

To solve this problem, many methods have been proposed, one of them is to classify by using another classifier to reinforce the main classifier (Ennaji et al., 2012). The author uses the Support Vector Machines (SVM) to reinforce the Self-Training model based on the KNN classifier. Compared to the normal Self-training based on KNN, this strategy has improved the classifier performance by 1% to 3%.

Another thing that caught our attention is that different feature representations bring different results. Most algorithms perform better on the Embedding dataset, only GNB performs better on the TFIDF dataset. The reason why TFIDF representation does not perform well under most algorithms may be that the frequency of words is not good enough to determine whether the sentiment is positive or negative. Negative emotion can be expressed without the use of negative words, for example, "How could anyone let this happen?" This sentence does not contain any obvious negative words, but we can say that it is a negative emotion. TFIDF representation may be more suitable for topic classification instead for sentiment classification.

6 Conclusions

In this paper, we demonstrate of different algorithms for the sentiment classification of tweets under different machine learning paradigms. From the experimental results, we found that when the prior knowledge is sufficient, the supervised learning algorithm outperforms the unsupervised learning algorithm in tweet sentiment classification. We also tried to use unlabeled data to improve the performance of the classifier, but since the classifier cannot guarantee that the automatically

generated labels are error-free, the experimental results are not optimistic. Fortunately, there are many methods that have been proposed to solve this problem, one of which is to use another classifier to help the main classifier make decisions (Ennaji et al., 2012), which can slightly improve the classifier accuracy.

References

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. (2016). *Demographic Dialectal Variation in Social Media: A Case Study of African-American English*. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130, Austin, Texas. Association for Computational Linguistics.

Ennaji, A., Mammass, D., & Yassa, M. E. (2012). Self-training using a k-Nearest Neighbor as a base classifier reinforced by Support Vector Machines. International Journal of Computer Applications.

Jiang, L., Yu, M., Zhou, M., Liu, X., & Zhao, T. (2011). Target-dependent Twitter Sentiment Classification. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 151–160. https://aclanthology.org/P11-1016

Kalaivani, P., & Shunmuganathan, K. L. (2013). Sentiment Classification of Movie Reviews by Supervised Machine Learning Approaches. Indian Journal of Computer Science and Engineering.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nédellec & C. Rouveirol (Eds.), Machine Learning: ECML-98 (Vol. 1398, pp. 4–15). Springer Berlin Heidelberg. https://doi.org/10.1007/BFb0026666

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. ArXiv:Cs/0205070. http://arxiv.org/abs/cs/0205070

Rothfels, J., & Tibshirani, J. (n.d.). Unsupervised sentiment classification of English movie reviews using automatic selection of positive and negative sentiment items. 7.