Abstract

Six Different ML Model on Tweet Sentiment Analysis

Chenqi Guo, Guoping Li, Ziqin Xu
chenqiguo2019@u.northwestern.edu; guopingli2017@u.northwestern.edu;
ziqinxu2017@u.northwestern.edu
EECS349 Machine Learning
Northwestern University

Our task is to predict the sentiment (positive or negative) in a tweet text dataset based on the text using several different ML algorithms, and finally compare the result and accuracy among these different algorithms.

More specifically: In this project, we analyzed the sentiment distribution and tried to recognize the sentiment features from test based on tweet sentiment analysis dataset. We plan to adapt different ML algorithms and visualize their performance under different testing conditions.

Why is the task important: Because sentiment analysis is useful in social media, like determining market strategy, improving customer service, testing business KPIs and so on. Also it is convenient to get the related datasets on Github, Kaggle or Twitter.

The models we used contain Logistic Regression, Support Vector Machine, Decision Tree, Nearest Neighbor, LSTM and Naive Bayes. The strategy we used to convert text into high-dimensional vector is called *bag of word* method, so the **features** are words in the word vector. To find a best representation of tweet text, we tried different dimensions of word vectors.

Key results:

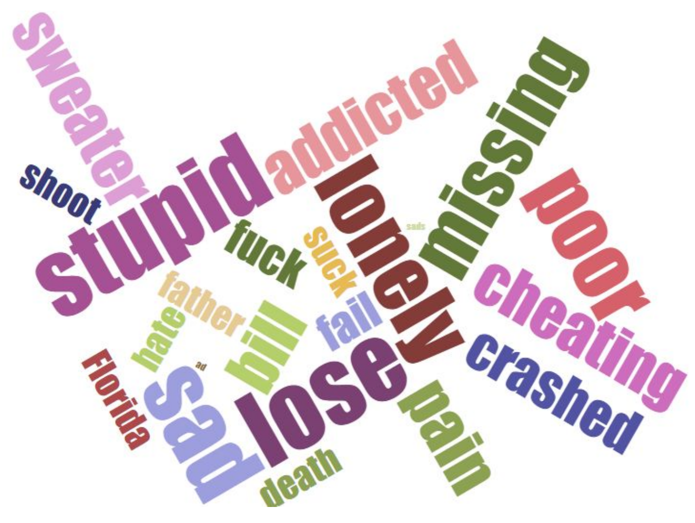
1. The effect of different bag of word

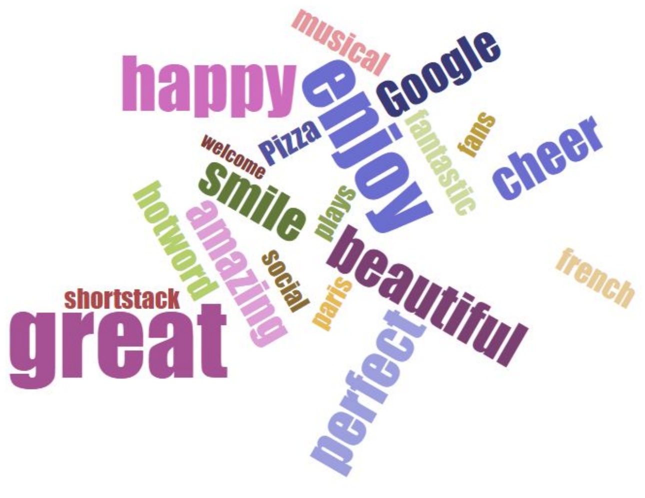
Generally, the accuracy of using a 1000-dimensional bag of word is higher than that of 2000-dimensional bag of word, because 2000-D increases the noise.

2. The effect of different learners

LSTM has the highest accuracy (about 75%), while Naive Bayes has the lowest accuray (about 66.5%).

Word Cloud using Logistic Regression shows the top 20 weighted words in positive sentences and negative sentences, respectively:





Detailed Final Report

Six Different ML Models for Tweet Sentiment Analysis

Chenqi Guo, Guoping Li, Ziqin Xu
chenqiguo2019@u.northwestern.edu; guopingli2017@u.northwestern.edu
EECS349 Machine Learning
Northwestern University

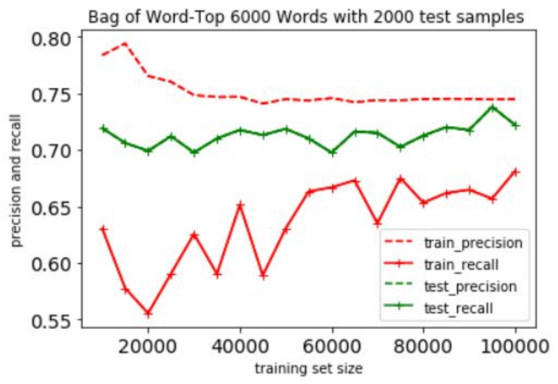
In this course project report, we discussed the difference of performance for 6 different ML algorithms, including Logistic Regression, Decision Tree, Nearest Neighbor, Long Short Term Memory, and Naive Bayes Classifier, and Support Vector Machine. We mainly explore the influence of training size to the precision and recall of the model. And we used the *bag of word* method to convert a sentence into a high-dimensional vector (6000, 4500, 3000, 2000, and 1000). The reason we used precision and recall to represent our results instead of the value of accuracy is because the dataset is not evenly distributed with positive and negative results.

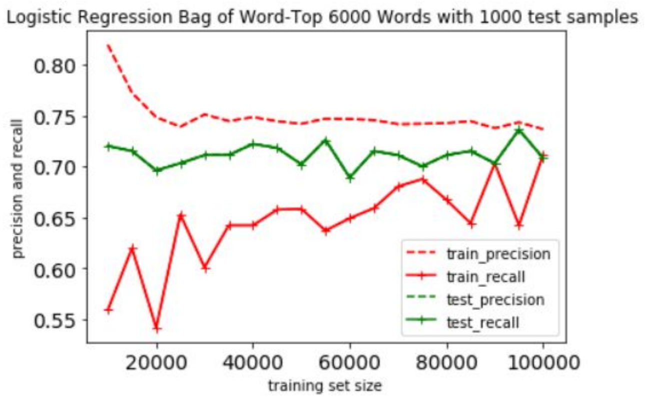
In the Logistic Regression, we first randomly chose 500 positive samples and 500 negative samples as testing set. The testing set will never be used for training. Then, we adapted different size of training set to train the Logistic Regression model. We can obviously find that the precision and recall value changed as we increased the size of training set. The line is fluctuating but the general trend is going toward down first and then reaching a plateau, which means the training set is gradually achieving the upper limit.

We then explore the influence of vector dimension. We used different dimensions (600, 4500, 3000, 1500) of vector to train the model. We found that no matter what dimension it is, all of them will finally achieve a plateau when the size of training set is large enough. However, significantly, we also found that a smaller vector size will result in a faster speed to achieve the performance plateau (Figure 1. red dash line).

Then, we decided to explore the influence of the size of test set. We chose 6000-dimension vector because we we would like to include as many features as possible. We found that more samples in the test set result in a more significant difference between test recall and train recall. We are still not very sure about the reason behind this tendency.

1. Logistic Regression





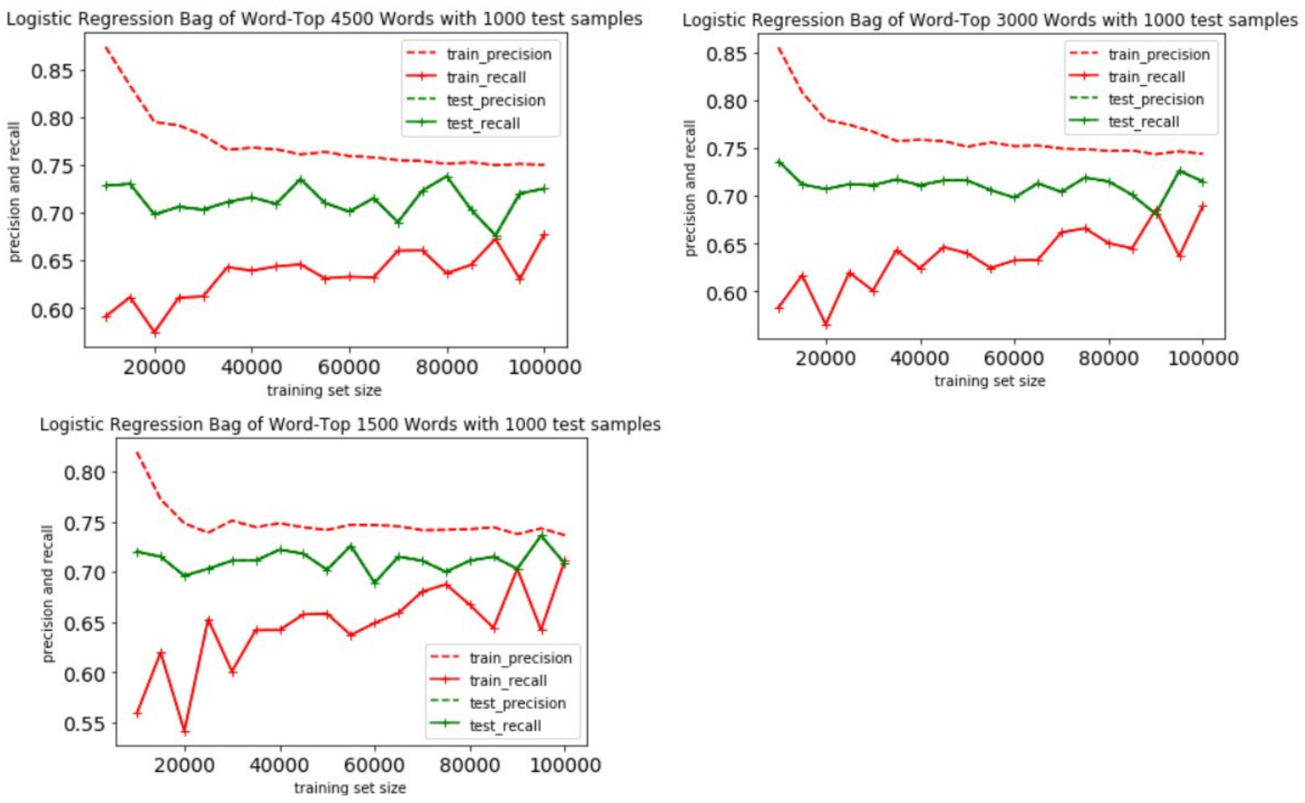


Figure 1: Logistic Regression using different bag of word and training size

2. Decision Tree
To evaluate the performance of difference classifier, we also used some classic algorithm like Decision Tree and Nearest Neighbor, as shown in Figure 2 and Figure 3.

For Decision Tree, since its VC dimension is infinite, we can get about 100% accuracy for the training data set. However, as we do not use a pruning strategy (the scikit learn package does not support pruning), this method is definitely overfitting when applied to testing data set.

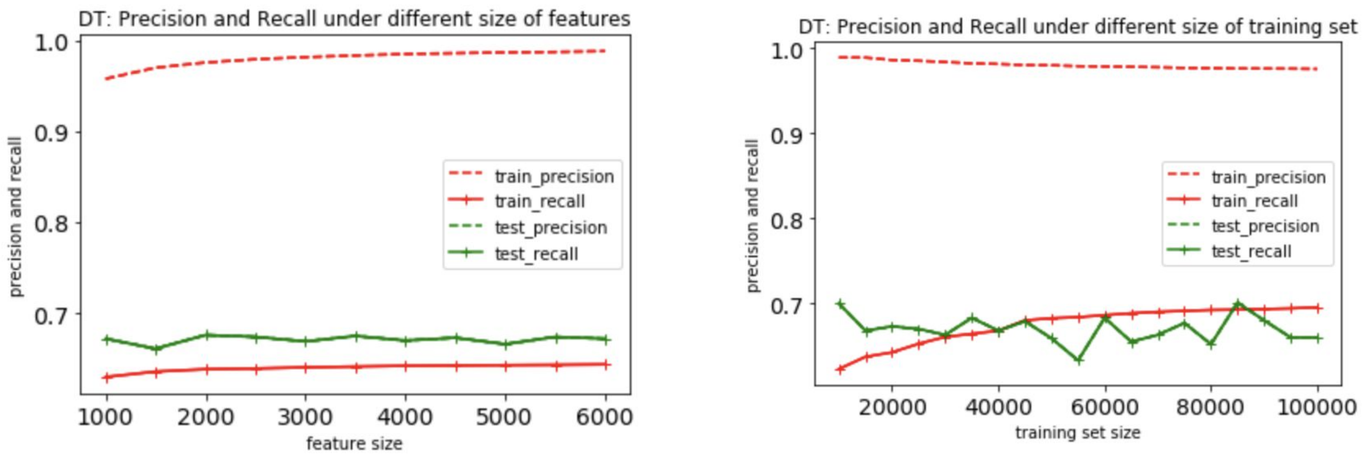


Figure 2: Decision Tree under different training set size and feature size

Figure 2 shows the effect of different feature size (with fixed training set size of 20000) and training set size on the precision and recall of training and testing data set. When applying different feature size (i.e., different bag of word), the training precision slightly increases, while testing precision and recall rarely change. When training our decision tree with increasing data set size, the training precision shows a slight decreasing trend while the test precision and recall fluctuate but still remain about 70%. Namely, feature size and training set size can hardly influence the testing results in decision tree.

3. Nearest Neighbor

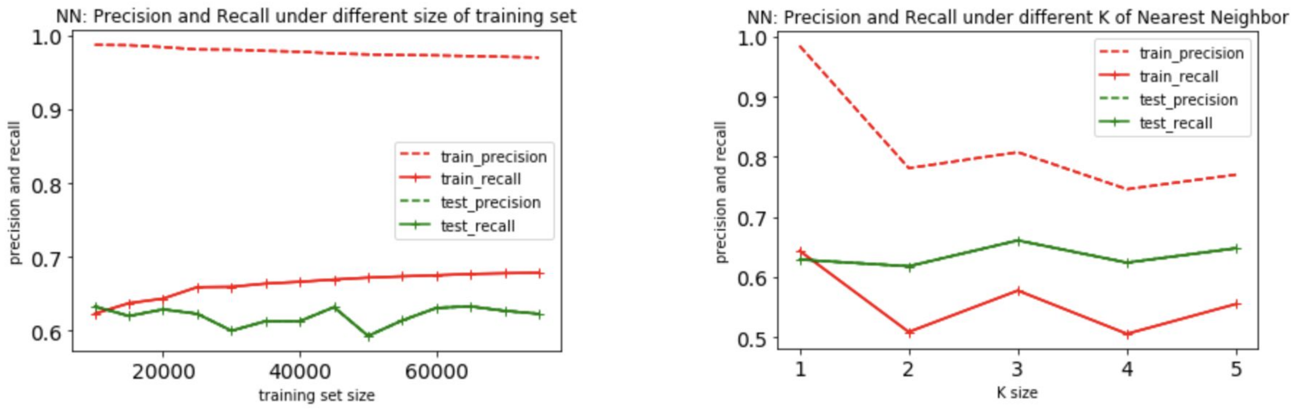


Figure 3: K Nearest Neighbor with different training set size and K value

Figure 3 shows the results we get with KNN algorithm. Also, since the VC dimension of Nearest Neighbor classifier is infinite, we can still get an 100% accuracy for training.
The figure in left shows 1-NN results on different training set size, while the figure in right shows results on fixed 2000 training data samples with different K values. As we increase the training set size, testing accuracy and recall fluctuates because of a trade off between informative data samples and noised features. As we increase the value of K, it is impressive that training precision decreased while testing precision and recall remained about 63%. A probable reason for this performance is that in 1-NN, each training data sample is the nearest neighbor for itself, but when K is greater than 1, each training data sample should consider others.

4. LSTM

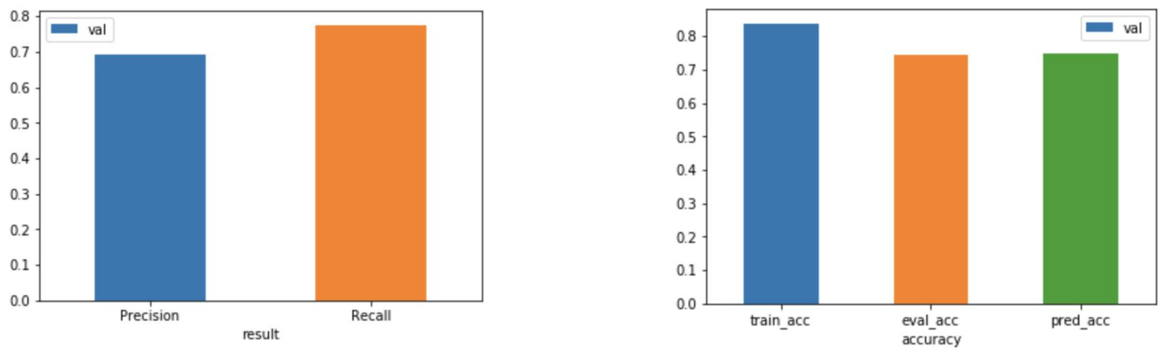


Figure 4: LSTM training, validation and testing result

In LSTM, because of the huge computation required to build word embedded layer and LSTM layer. We just used 10000 tweets to build one model and output test result as a histogram. Here we can see recall is high as about 0.75 and precision as 0.68 which are not so good but pred_acc as test accuracy is good enough as 0.71. Also, variance is not so large as about 0.1 so overfitting is not a big problem in this model. Ideally, with larger dataset input will show the true power of this LSTM model with less bias and improve precision to a much more reasonable result.

5. Naive Bayes

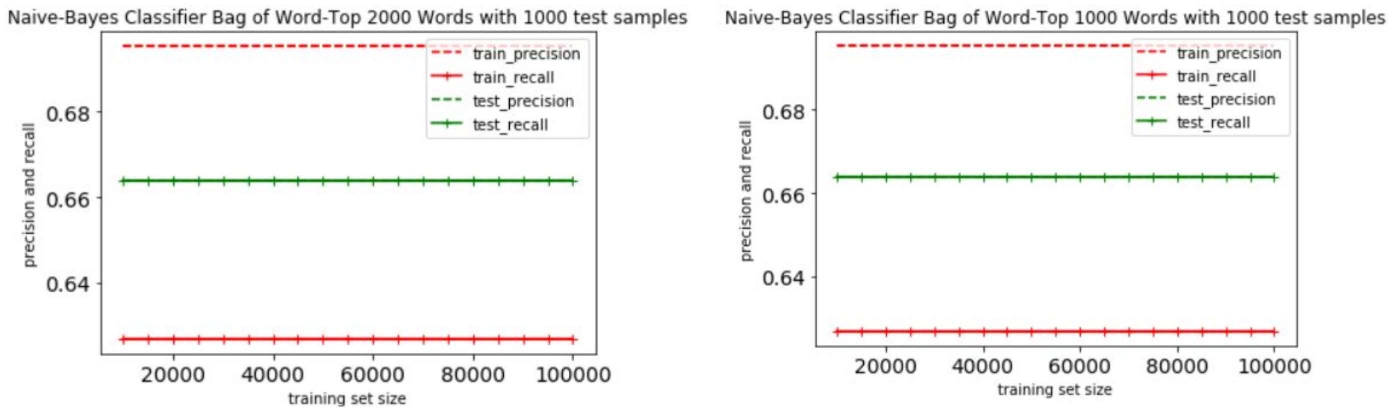


Figure 5: Naive Bayes classifier under different training set size and feature size

6. Support Vector Machine

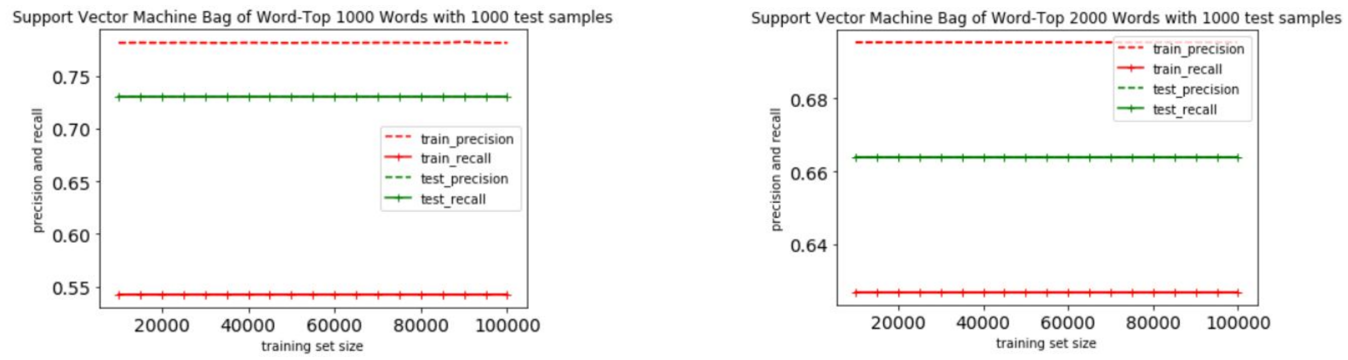


Figure 6: Support Vector Machine under different training set size and feature size

In the Naive Bayes classifier and support vector machine classifier (Figure 5 and Figure 6), each the length of sentence is much less than the dimension of vector. So, Nearly all of the vectors are very sparse, which created a challenge for linear support vector machine classifier and naive bayes classifier. Because in Naive Bayes classifier, we did not apply the Add One Smooth strategy to avoid the risk of getting zero. So, in these two classifiers, they did not catch the main features words at all. Their performance is irrelevant to the size of training set. Their precision and recall value stay at a constant value for whatever size of the training set. So the Support Vector Machine classifier is not very effective when handling a very sparse vector. And we learned that we should apply the Add One Smoothing method to the Naive Bayes classifier before we initialized the training. More importantly, we should develop and test other vectorization methods instead of the *bag of word* when using these two ML models.

Conclusion:

In this project, we explored the performance of 6 different ML algorithms for sentiment analysis. We found that given the *bag of word* method, the Logistic Regression algorithm has the best test precision value (0.75) when the dimension of vector are 1500 and 3000 and the size of training set are around 90,000. Then, we extracted the words with highest weight classified by Logistic Regression algorithm, and we draw two word clouds for the positive words and negative words, respectively. (Figure in abstract, a bigger size of word in a word cloud means this word has a higher weight)

The Decision Tree classifier shows a nearly perfect training precision, but the test precision and test recall are lower than any other algorithms we explored. This is because the decision tree classifier is extremely sensitive to overfitting.

The Nearest Neighbor classifier also provide great training precision. But its test precision and test recall is close to the performance of decision tree classifier. And we tested its performance when we have different values of K. It shows that the test precision is kind of irrelevant to the value of K, when K is in range of 1 to 5. This might due to the feature of *bag of word* method, because the bag of word method created a very high-dimension vector, which involves lots of noise to the L2-distance of K-NN algorithm.

Technically speaking, LSTM may have a very competitive test precision (0.68) among all these ML methods we used, since it is a refined version of neural network, which can perform well in this task. However, due to the limitation of our laptop's RAM and GPU, we can only get an accuracy as good as the best one among other methods in practice. We will try to fix this memory problem with Google Cloud in the future.

Finally, we also found that the extent of sparseness of vector is critical for some classifiers,including Naive Bayes classifier and Support Vector Machine classifier. If the vector are sparse, the classifier lose its capability to classifier because those 0 values would cause a big bias, which makes the classifier's prediction become random. Even in this situation, the NBC and SVM still obtain a test precision around 0.65, which is close to the performance of DT and K-NN.

Cooperation:

In this project report, we worked as a team. The construction of web pages were completed by Ziqin.The abstract part of this report was mainly completed by Chenqi and Guoping. All of the datasets using in this project were downloaded from public open source (twitter, kaggle, etc) and were process to remove punctuations, stop words and numbers by Guoping and Ziqin. The detail final report and the corresponding codes with visualizations are completed by Guoping (Logistic Regression, Support Vector Machine, Naive Bayes classifier, figures of word cloud and the conclusion part), Chenqi (Decision Tree and Nearest Neighbor) and Ziqin (LSTM).