Sentiment Extraction Of Twitter Messages

Hanyuan Zhang, Liqiang Luo, Tinglong Liao
New York University Shanghai

Abstract

In the current decade, we witness the booming of social media, people all around the world express their feelings and emotions on social media. In this project, we analyzed 31026 twitter messages and focused on capturing the sentiments of those tweets. We first discuss different feature extraction methods, including One-hot, TF-IDF, Word2Vec and GloVe. Then, we apply various models with above feature extraction methods, including Naive Bayes, Logistic Regression, Ada-Boosting, SVM and LSTM. The result was not perfect, we reached the highest accuracy rate of 75.43% by using LSTM with Glove. This comparatively low accuracy could attribute to the small size of our dataset. And we believe that in the future, when a larger amount of data is collected, a significant higher accuracy rate could be obtained.

In addition, we find that in traditional non-neural-network models, feature extraction methods have a more significant impact on performance than different models. This could be indicative to our future work that more effort could be put on finding better feature extraction methods.

I. Introduction

Millions of tweets are circulating every minute. If we can capture the sentiment of tweets, we can conduct a variety of further data analysis. For instance, together with the tag and location of each tweet, we can analyze how people react towards a certain event. Take the outbreak of coronavirus as an example, we can answer the following questions: when the virus first broke out, how did people around the world perceive it? Are the majority of them worried, or just kept cool and watched what on? Did people in the United States feel more indifferent than people in Japan? As time went on, did people's emotions change towards the virus? We can also imagine tons of other possible applications of sentiment analysis of tweets: use the timely emotion as a reference for advertising, draw a profile for each user based on his or her average emotion, and so on. We believe that capturing the sentiment of tweets is fun and has applicable value, which motivates us to choose to analyze the emotion of a tweet as our final project theme.

In our report, we will first introduce our dataset. After that, we will explain different feature extraction methods we use. Then, we will introduce different models we use. We will explain the reason we choose them, show their performance and analyze their limitations. Finally, we will provide an discussion of the result and future development.

II. Dataset

The dataset that we used comes from Kaggle**[1]**, which collected 31026 Twitter posts and labeled the sentiment of each tweet as negative, neutral, or positive.

In the original dataset, 40% of the tweet texts are labeled as neutral, 31% labeled as positive, and the remaining 29% are labeled as negative(shown in Figure 1). This is not a strictly balanced dataset with respect to sentiment distribution, and would theoretically benefit from randomly dropping some neutral text to make the percentage equal. However, the size of our dataset is comparatively small, thus downsizing the neutral class would cause more problems than benefits.

We also analyze how the number of words in the Twitter messages and the distribution can be seen in Figure 2. Overall, the length of tweets is between 1 word and 31 words(separated by space). And they center around 5-10 words. From this, we can tell that the length of the text in our dataset is relatively short. This is consistent with our everyday experience where people tend to post short tweets.

We randomly split the dataset into training and testing data by an 80-20 proportion. And then split the training data into training and validation data by an 80-20 proportion. As a result, we got 19857 train data, 6205 test data, and 4964 validation data.

III. Method

A. Feature Extraction

1) One-hot: A one-hot vector is a 1 × N matrix used to distinguish each word in a vocabulary from every other word in the vocabulary. The vector consists of 0s in all cells with the exception of a single 1 in a cell used uniquely to identify the word. It is used to convert our text into numeric form which contains the frequency of each word in vocabulary.

2) TF-IDF: Term frequency–inverse document frequency (TF-IDF) is a numerical statistic that is intended to reflect how important a word t is to a document d in a corpus D.

$$\textbf{TFIDF}(t, d, D) = \textbf{TF}(t, d) \times \textbf{IDF}(t, D)$$

where

$$\textbf{TF}(t, d) = \frac{\text{number of times } t \text{ appears in } d}{\text{number of word in } d}$$

$$\textbf{IDF}(t, D) = \log(\frac{\text{number of } d \text{ in } D}{\text{number of d that contains } t \text{ in } D})$$

A term will have a high TD-IDF weight if it has a high term frequency in the tweets and a low document frequency of the term in the whole collection of tweets.

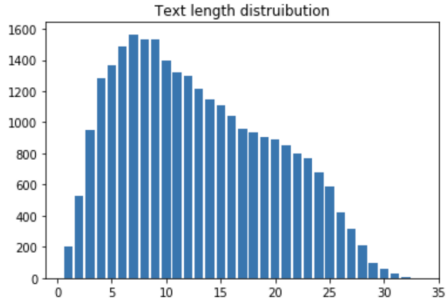

Figure 1. text length distribution

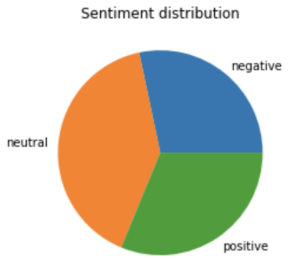

Figure 2. sentiment distribution

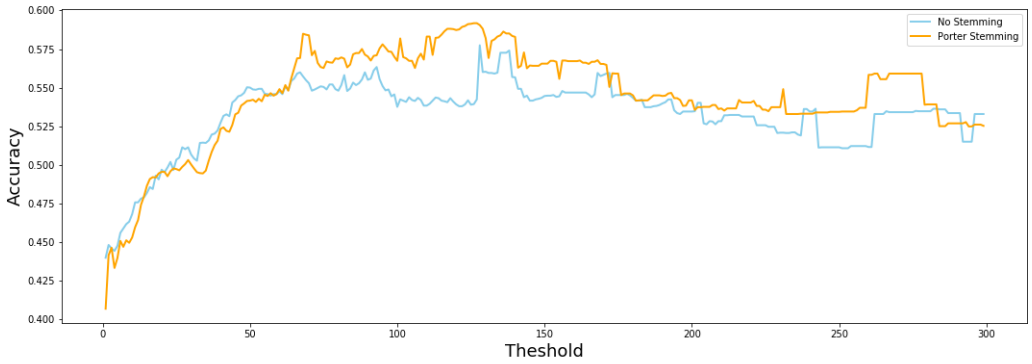

Figure 3. Naive Bayes

3) Word2Vec: Word2Vec represents each distinct word with a particular vector. The vectors are chosen to ensure words with similar semantic meaning have smaller Euclidean distance or cosine similarity. In our model, we apply gensim Word2Vec model to train our word vectors base on sentences in our training set.

4) GloVe: GloVe is a pre-trained Word2Vec. The advantage of GloVe is that, unlike Word2vec, GloVe does not just rely on local statistics, but incorporates global statistics like corpus from Wikipedia to obtain word vectors. Here we use the pre-trained word vectors by Stanford NLP team as our word embedding vectors**[2]**.

B. Model

1) Naive Bayes: We apply Naive Bayes based on one-hot encoding. The Naive Bayes model is easy to implement because only the probability is to be calculated. However, under the context of sentiment analysis, it is obvious that each word in one sentence is correlated with other words. Hence, the independence assumption does not hold and the model's performance could be quite low. Therefore, we are only using Naive Bayes as our baseline.

For the Naive Bayes model, first we tune minimum document frequency. When creating vocabulary, the algorithm will ignore terms that have a document frequency strictly lower than minimum document frequency. As is shown if Figure 3, the accuracy first increases and then decreases. One reasonable explanation is that when minimum document frequency is too low, the noise of those rarely occurring and misspelling words would influence the model performance. And when the minimum document frequency is too high, we are losing information in our features.

We also carry out word stemming. The performance becomes better to some degree but the accuracy is till very low. (see in Figure 3)

2) Logistic Regression: Logistic regression makes no assumptions about distributions of classes in feature space. It is easy to implement and fast to train. Also, we can interpret model coefficients as indicators of feature importance. In sentiment analysis, if we are using one-hot encoding or TFIDF, we can easily check which words are attached with the highest weights.

However, it is very possible that our data is not linearly separable. It is tough to obtain complex relationships by logistic regression.

First, we try logistic regression with TF-IDF. We tune 2 hyperparameters.The first hyperparameter is the vocabulary size. Given a satisfactory result, we want to keep the size of vocabulary as small as possible to make the program run faster. The second hyperparameter is maximum document frequency. When building the vocabulary, the algorithm ignores terms that have a document frequency higher than maximum document frequency. Before tuning, we think that a small maximum document frequency can help us to get rid of those too

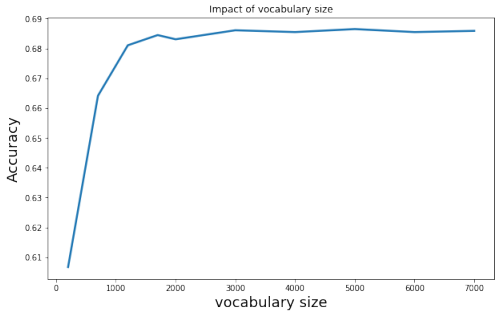

Figure 4.

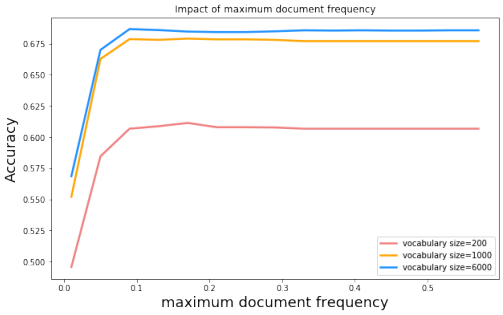

Figure 5.

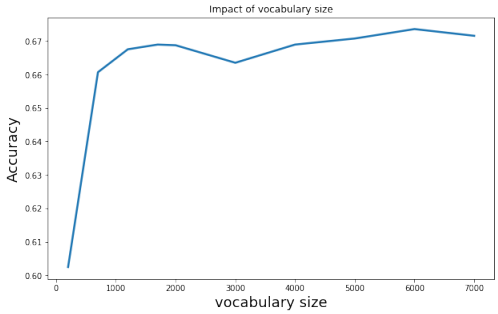

Figure 6.

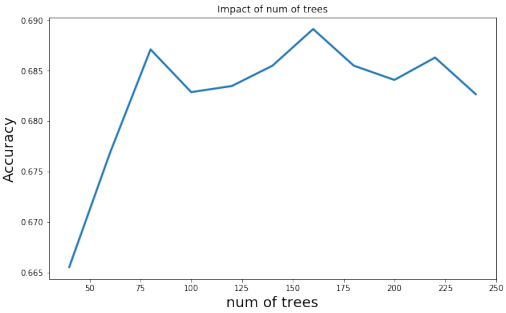

Figure 7.

common words like “a” , ”the” and so on, which don’t help with the classification.

It turns out the model performance improves tiny when vocabulary size > 3000. Also, a small maximum document frequency is not helpful. Setting vocabulary size = 5000 and maximum document frequency = 0.5, the accuracy on the test set is 69.5%. (see Figure 4 and 5)

We also try logistic regression with Word2vec, Glove. For each sentence, we take the average of its words vector as its sentence vector. We also try to concatenate sentence vector of Glove and sentence vector of TF-IDF, the result is as follows:

	Word2vec	GloVe	TF-IDF + GloVe
Accuracy	47.41%	60.60%	68.53%

3) Ada-Boosting: Adaboost aims at combining several weak learners to form a single strong learner. Any observations that were classified incorrectly are assigned more weight than other observations that had no error in classification. An error noticed in previous models is adjusted with weighting until an accurate predictor is made.

Firstly, we try ada-boosting with Glove. For each sentence, we take the average of its words vector as its sentence vector. After training, its accuracy is around 59.63%. We also try ada-boosting with TF-IDF vectors. Similar to logistic regression with TF-IDF, We also tune vocabulary size and maximum document frequency. (see Figure 6 and 7)

Taking max features = 6000, and n_estimators = 140, the accuracy on the test set is 68.22%.

4) Support Vector Machine: Since it cannot be avoided that our features always have high dimensional spaces, we decide to try to use Support Vector Machine (SVM) as our model and fit the model with TF-IDF.

One of the top advantages of using SVM is that, it is more effective in high dimensional spaces. So it may have a better performance when we fit it with high dimensional spaces features. SVM model can also handle the classification with nonlinear boundary by setting a kernel, which is probably helpful for our sentiment classification. The accuracy of our model reaches 70% as we increase the threshold of TF-IDF. (see Figure 8)

However, the runtime of SVM model is much longer compared with Logistic regression. In order to make our training process more effective, we also try SVM with stochastic gradient decent as a balance. The runtime decreases significantly and the accuracy stays around 64%. (see Figure 9)

5) LSTM: The main drawback of the previous models is that those models completely disregard the effect of the order of the words within a sentence. So it’s reasonable to use an LSTM model to tackle this problem.

In our neural network, we have one embedding layer, one two-layered, bidirectional LSTM layer and one fully connected layer with dropout. The final output is activated by a softmax function and we chose Adam to be the optimizer. One thing to notice is that in LSTM layer, we chose the hidden state before reaching the padding to avoid the end paddings from damaging the model performance.

There are a number of parameters to tune: input dimension, hiddenstate dimension, layers of LSTM layer, bidirectional LSTM or single direction, and what to output in the hidden state etc, but none of the above showed a significant impact on accuracy. As the result, we reached 75.42% accuracy when running this model.

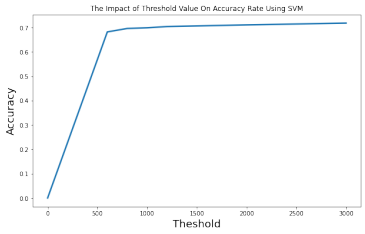

Figure 8.

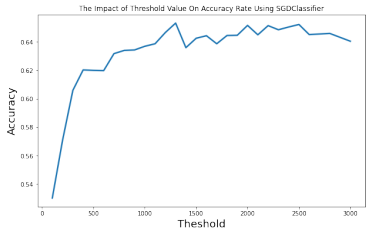

Figure 9.

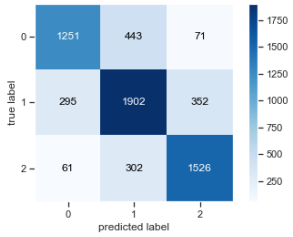

Figure 10.

	One-hot	TF-IDF	Word2vec	GloVe
Naive Bayes	57.68%	-	-	-
Logistic Regression	-	69.50%	47.41%	60.60%
Ada-Boosting	-	68.22%	-	59.63%
SVM	-	71.84%	-	60.87%
LSTM	-	-	-	75.43%
Random Guess			33.33%	

Figure 11. result table

Due to the fact that the size of our data is relatively small, we keep the neural network quite simple to avoid over-fitting the training data. In fact, in our experiment, most models' validation lass start to increase after 4-5 epochs while train loss continues to decrease. This means that even a simple LSTM model like ours provides certain over-fitting phenomenon.

IV. Result and Discussion

As shown in the result table, Naive Bayes performs better than a random guess. But its accuracy is not satisfactory. Logistic regression, Ada-Boosting and SVM perform better than Naive Bayes. But none of them is able to consider the sequence of words within a sequence. LSTM with Glove performs the best and reaches the accuracy rate of 75.42%. This is consistent with our assumption that the order of words within a sentence is important to the sentiment of the tweet sentiments

A. Misclassification analysis

Here we take a close look at the misclassified texts to see what kind of texts are prone to being misclassified. Take the prediction of the LSTM model as an example, the confusion matrix is shown in Figure 10. As we can see, in regard to misclassification, LSTM model is not very likely to misclassify a positive tweet to a negative sentiment or vice versa, but often makes mistakes regarding neutral sentiment. This suggests that our model is on the right track, and not making random wrong assumptions. When analyzing the misclassified data, we recognized two common patterns of misclassified data. Firstly, important phrases that support the sentiment can be replaced by unknown tags. For example, "So cooooool" is an obvious positive tweet. However, the o in word cool is repeated many times and being recognized as unknown. Therefore, the model only sees "so <unk>" for the sentence and would misclassify it as a neutral tweet. Secondly, there are many tweets that are vague in sentiments even for humans. For example, "It's because you are popular" can be a neutral answer or a positive praise. Therefore, it's reasonable to allow our models to make mistakes of this kind.

B. Limitation of feature extraction method

Also, we find that in traditional non-neural-network models, feature extraction has a larger impact on performance than different models. For instance, when using TF-IDF, Logistic Regression, Ada-Boosting and SVM share similar accuracy rates. But there is a huge gap between performance of SVM with TF-IDF and SVM with GloVe. Hence, we believe that it is the feature extraction that limits the performance of our models.

When applying TF-IDF on tweet sentiment analysis, term frequency could easily lose its effectiveness. The text content of a Tweet can contain up to 140 characters. In our training set, The longest sentence only contains 31 words. Under such context, it is rare to see repeated words in one tweet, except meaningless words like "a" and "the". Also, TF-IDF is not able to consider the sequence of sentences. The above reasons explain the limitation of applying TF-IDF here.

For Word2Vec, it usually malfunctions to separate some opposite word pairs because the distance between different words only represent the probability of them showing up together. For example, "good" and "bad" are usually located very close to each other in the vector space which may limit the performance of sentiment analysis.

V. Conclusion

In conclusion, we tried different methods to analyze the sentiment of tweets. Frist, we discuss different feature extraction methods, including One-hot, TF-IDF, Word2Vec and Glove. Then, we apply various models with above feature extraction methods, including Naive Bayes, Logistic Regression, Ada-Boosting, SVM and LSTM. The best model we develop is LSTM. LSTM is the only model to be able to consider the influence of sequences of words. It has 75.43% testing accuracy. Interestingly, we find that in traditional non neural network models, feature extraction methods play a more important role than classifiers. In the future, more effort should be put on finding better feature extraction methods and finding more data to train our model.

Reference

[1] Kaggle Dataset: https://www.kaggle.com/c/tweet-sentiment-extraction

[2] GloVe: https://nlp.stanford.edu/projects/glove/