People' s Daily Micro-blog Simulator: Applications of Machine Learning Models to Assist Editors to Evaluate Post Performance

Anran Wang, Tian Jin, Zhenming Wang

Abstract

Weibo is one of the largest social platform where tens of millions of users publish and exchange information everyday. Among the many official accounts, *People's Daily* is one of the most influential ones. Therefore, it is worth looking into the text corpus of *People's Daily*'s weibo account and is insightful to explore the relationship between each blog post and their popularity on Weibo, under the measurement of likes, comments and forwards.

We used several machine learning models to address our research problem, from linear regression to neural networks. In general, support vector machine out perform other models with the smallest testing error. With the user interface we build for people to type in the and give simultaneous prediction on the number of forwards, likes and comments, this interface could actually help the user better adjust the content to create a more popular post.

1 Introduction

Weibo is one of the biggest Chinese social networks with more than 200,000,000 Daily Active Users [1]. With over 38,000 official accounts operating, ranging from government departments to media presses, Weibo gradually becomes a source of information. Therefore, it' s a lively digital database for researchers to study the propagation of information on Chinese Internet.

Among all of the official accounts, *People' s Daily* is quite influential. As the largest newspaper agency representing the government, *People' s Daily* wins over 100,000,000 followers on Weibo. Each of its posts gains hundreds or thousands of comments, likes, forwards, while expressing different kinds of emotions. In early 2020, news about COVID-19, its current development and global trends always caught much attention. This might give us some insights into how the content of the post might be related to its popularity.

Our research question is then: given a piece of weibo blog post, what can we say about its possible popularity on weibo, in terms of number of likes, forwards and comments it is going to receive?

Previous works studying weibo posts with machine learning techniques exists. However, most of them focused on extracting sentimental information from the text[3, 4], which is drastically different from our research. Our study primarily utilizes texts of blog posts as feature vectors to predict numerical values including the number of forwards, likes and comments.

2 Dataset

a Data Source

The dataset we used for this study is a collection of approximately 130,000 posts from the official Weibo account of *People' s Daily.* The publication date of the posts ranges from Jan. 2012 to Apr. 2021 in our collection. We build our own data-retrieving scripts with the help of external web crawler library [weibo spider], and the process of retrieving and storing all posts took about three days.

The data extracted from weibo is structured, with each field of content stored as a column in a .csv file. This includes weibo id, weibo text, url, publication date, number of forwards, likes and comments, etc. Weibo text contains the actual content of each blog post. Number of forwards, likes and comments are numerical measurements. For our study, we only focused on these four columns: weibo text, number of forwards, likes and comments.

b Data Pre-processing

We performed necessary data pre-processing steps. Since our dataset comes from weibo, it is in Chinese by default, we have to specifically apply Chinese Natural Language Processing algorithms. First, we remove meaningless symbols from the original texts using regular expression. Secondly, we use a python library "jieba" to tokenize[1], separating full sentences into separate words. Lastly, we convert the outcome of jieba into sentences by concatenating the word with spaces.

Regular expression: At first, since python library "jieba" can tokenize, we directly apply "jieba" to convert full sentences into words. However, "jieba" tends to combine meaningless symbols, such as "#", "【" with other words, albeit it supports stopword. Since "jieba" could not get rid of the stopwords, we decide to use regular expression so that symbols, punctuations, interjections will be removed. We apply a filter **"\w+"** to pick all words and numbers ranging from 0 to 9, and all other characters are discarded.

Tokenization: One major difference between Chinese and English corpus is that English sentences are naturally separated by spaces, thus it's simpler to convert English corpus to collections of words. In comparison, Chinese sentences have no boundaries for words. Here we utilize "jieba", which is a powerful Chinese tokenization library. It supports deep learning framework to separate words with high precision. By applying "jieba" on the pre-processed text we obtained, full sentences are broken down to words.

Word Concatenation: After tokenization, one single sentence is now a list of separated words. We concatenate each word within the list again, now adding a space in between. In this way, the sentences is ready to be transformed into vector representations.

Log Transformation: There are three numerical values in our dataset: the number of forwards, likes and comments. As we plotted out the distribution of all data points, we found that it is very skewed and approximately follows a log-normal distribution, with about 50% of the data concentrating in the first bin. Figure 1 shows the original distribution of the number of *likes*.

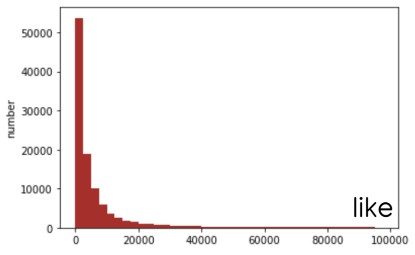


Figure 1: likes values before processing

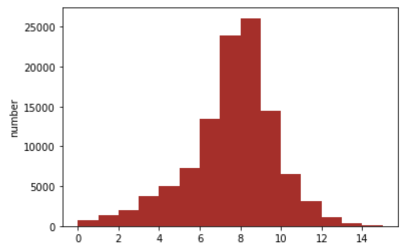


Figure 2: likes values after processing

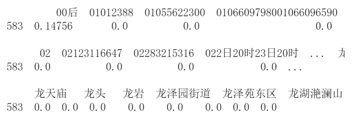
To reduce the skewness of the original data such that following statistical analysis could be more valid and applicable, we applies log transformation the number of forwards, likes and comments. After transformation, the distribution follows approximately a normal distribution, as is displayed in Figure 2.

3 Methodology

a Feature Extraction

1) *TF-IDF*
Apparently, if we want to input the text data into machine learning models, we have to transform text data into feature vectors. In this study, we choose *TF-IDF*.

TF-IDF offers a statistical measure that eval-



```
        00后   01012388   01055622300   0106609798001066096590
583  0.14756        0.0           0.0                        0.0

       02   02123116647   02283215316   022日20时23日20时   ...   龙
583  0.0             0.0           0.0                0.0  ...

     龙天庙    龙头    龙岩   龙泽园街道   龙泽苑东区   龙湖滟澜山
583  0.0  0.0  0.0    0.0     0.0     0.0  0.0  0.0
```

Figure 3: TF-IDF working example

uates how relevant a word is to a document in a collection of documents[2]. Each word appearing in a document get a score by multiplying two separate values, *term frequency* (tf) and *inverse document frequency* (idf). The term frequency of a word is decided by the number of times it appear inside one document. The more frequent the occurrence, the higher the term frequency. The inverse document frequency is decided by how common a word is across all documents. The more common a word is in the documents set, the closer to zero its idf value is, and vice versa. The higher the *TF-IDF* score, the more relevant that word is in that particular document[2].

Figure 3 is an example of *TF-IDF* value in our dataset. We implemented *TF-IDF* using *sklearn CountVectorizer* and *tfidfTransformer*. *CountVectorizer* transformed a set of documents into a matrix of word counts. There is a hyper-parameter *max_features* that comes along with *CountVectorizer*, which controls the number of words taken into account during the process of the study. In the end, we converted our preprocessed text corpus into a collections of vectors each of size *max_features* $\times 1$.

A worth mentioning point is that the output

2) *Mean Absolute Error*
When evaluating model performances, we choose *mean absolute error* (MAE) as the measurement. Because we are predicting the number of forwards, likes and comments on weibo, MAE signifies the amount away from ground truth value for each post, which is very straightforward and gives a clear sense of the model performance. In the scope of our study, as our original dataset is very skewed, MAE is less sensitive to outliers. If using Mean Squared Error (MSE), the outcome would be much greater and cause confusion when evaluating the model.

b Models

1) *Linear Regression*
Linear regression, as the most basic regression model, is the first model we tried. Here, we choose *max_features* = 10000 to be our input, so that only the top 10000 term frequency words' TF-IDF will be included. Because the dataset is too big and it is too time consuming to fit in more features. At *max_features* = 10000, the testing MAE is 5037.8056129 (forward), 12240.8485751 (like), 1173.41195107 (comment). The bad performance suggests that we might need more complex, non-linear models. Another possibility is that the choice of *max_features* is too small. In Figure 4, we tune the hyper parameter *max_features*. When more features are input into the model, the testing MAE will decrease, which suggests room for future improvement.

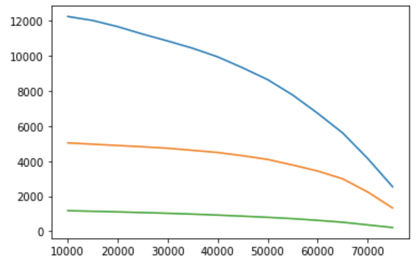


Figure 4: Tuning Max_features with LinearRegression

2) *Random Forest*
In order to introduce non-linear models into our tool box, we tried random forest. We tuned two hyper-parameters: *max_features* and *max_depth* of sklearn's implementation of random forest.

As *max_features* controls the dimension of input vectors, a smaller choice of number of words to keep when generating the tf-idf vector can significantly reduce training time. For example, running a random forest regressor with 150 estimators and *max_depth* of 10 under the setting when $max\_features = 30000$ took us 471 seconds; the same random forest regressor took 971 seconds when $max\_features = 100000$.

By tuning *max_depth*, decreases in both testing error and training error are seen. Figure 5 is a plot showing the decrease of both testing and training error for all of the three target variables.

As a conclusion, an average approximation of MAE for likes is around 11000; the average approximation of MAE for forwards is around 5000; the average approximation of MAE for comments is around 1100.

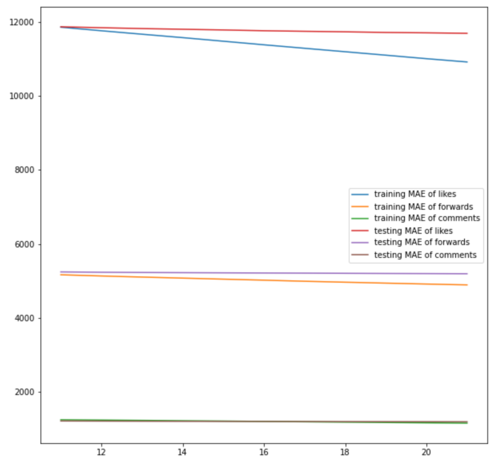


Figure 5: Random Forest: mean absolute error for both testing and training

3) *Support Vector Regression*
Support vector regression is another nonlinear model when we introduce the kernel. Because support vector regression also takes large runtime, we choose $max\_features = 10000$ to be our input. One hyper parameter we consider here is the type of the kernel, either rbf, polynomial or sigmoid. The testing MAE on the whole data set gives us the following result.

In comparison, rbf kernel performs the best for all predictions, with testing MAE , 1904.066555 (forward), 2383.823007 (like) and 957.821764 (comment). With same number of $max\_features$ input, SVR performs much better than linear regression. One possibility is that it is due to the collinearity between the our target variables. As the number of forwards, likes and comments could be related to each other, SVR solves collinearity by predicting one target variable at a time.

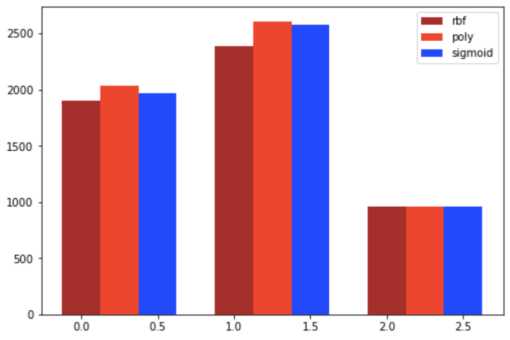
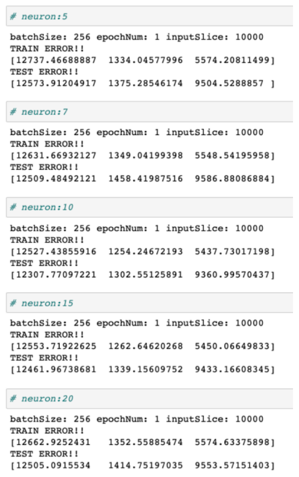


Figure 6: MAE with different SVR kernel

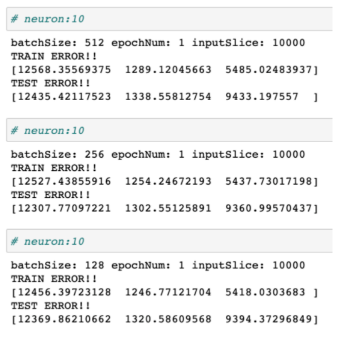
4) *Neural Network*
In order to solve the problem of not being able to feed in the whole large dataset, we also apply Neural Network using PyTorch. Specifically, we use torch.utils.data.dataset and torch.utils.data.dataloader to separate the original datasets into batches respectively, therefore making it possible to train the model with the entire dataset instead of only part of it. The number of neurons in the input layer is 10,000 corresponding to *max_features* and the output layer is 3 representing "like" , "forward" and "comment" .

As for the process of tuning parameters, we mainly apply the control variates method to tune the size of each batch and the number of neurons in the hidden layer. With other variables keeping the same, the best result comes when the batch size equals 256 (other trials include 64, 128, 512). Similarly, 10 turns out to be the best choice (from 5 to 35) of the number of neurons in the hidden layer.



```
# neuron:5
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12737.46688887  1334.04577996  5574.20811499]
TEST ERROR!!
[12573.91204917  1375.28546174  9504.5288857 ]

# neuron:7
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12631.66932127  1349.04199398  5548.54195958]
TEST ERROR!!
[12509.48492121  1458.41987516  9586.88086884]

# neuron:10
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12527.43855916  1254.24672193  5437.73017198]
TEST ERROR!!
[12307.77097221  1302.55125891  9360.99570437]

# neuron:15
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12553.71922625  1262.64620268  5450.06649833]
TEST ERROR!!
[12461.96738681  1339.15609752  9433.16608345]

# neuron:20
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12662.9252431   1352.55885474  5574.63375898]
TEST ERROR!!
[12505.0915534   1414.75197035  9553.57151403]
```

(a) number of neurons



```
# neuron:10
batchSize: 512 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12568.35569375  1289.12045663  5485.02483937]
TEST ERROR!!
[12435.42117523  1338.55812754  9433.197557  ]

# neuron:10
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12527.43855916  1254.24672193  5437.73017198]
TEST ERROR!!
[12307.77097221  1302.55125891  9360.99570437]

# neuron:10
batchSize: 128 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12456.39723128  1246.77121704  5418.0303683 ]
TEST ERROR!!
[12369.86210662  1320.58609568  9394.37296849]
```

(b) size of batches

Figure 7: Tuning parameters with Neural Network

The best result we get using one-layer Neural Network comes from the model with 10 neurons in the hidden layer and 256 training data utilized per iteration. We eventually train the model for 20 epochs on the entire dataset and get the MAE of 11446.5 (like), 5098.41 and 1116.83 (comment), which is better than Linear Regression and Support Vector Regression.

One possible improvement here lies in the amount of data that we include when tuning the parameters. Since it costs rather long time to train the whole dataset especially as we increase the number of neurons or size of batches, we only use 10,000 input out of over 100,000 when tuning the parameters. If there' d be infinite time, we will use the entire dataset to train the model for more epochs in order to get a more promising result.

4 Results and Discussion

1) *Model Performance*

In terms of the same max_features number, support vector regression outperforms other models, including one-layer neural network, linear regression and random forest. However, due to the limited time, the best performed model is actually linear regression trained on max_features =75000, which gives us testing MAE 1327.7118411 (forward), 2534.55446626 (like), 213.03970434 (comment). We consider the reason is that with more max_features input in the model, the testing MAE will improve with a higher rate. This suggests that if we train SVR on max_features =75000 or more than 75000, it will yield the best result.

2) *User Interface*

We deployed our linear regression model on a simple python server, and built a weibo-style web interface where user can input a new piece of post. Once user decide to publish their posts, it will be sent to a pre-trained machine learning model on the server and predicted value of likes, comments and forwards get generated, and get outputted on user's screen.

5 Future Improvements

One of the most challenging part of our project lies in the size of our original dataset. As we have more than 130,000 input data and each is converted into a 10,000 *max_features* feature matrix, it vastly influences the run time of all models and we fail to plug in more features for the input. If time permit, we could fit every model with max_features = 100000, anticipating that the testing MAE would be greatly improved. Also, previously, we're thinking about implementing sentiment analysis, giving additional an index for the emotional tendency of user responses. In this way, we might be able to also generate the general emotional response tendency.

References

[1] fxsjy. *jieba*. `https://github.com/fxsjy/jieba`. Accessed on 2021-5-15.

[2] Bruno Stecanella. *What is TF-IDF?* `https://monkeylearn.com/blog/what-is-tf-idf/`. Accessed on 2021-5-15. May 2019.

[3] 罗凌 吴昌 陈毅东 史晓东 曹茂元. “厦门大学 NLPCC 2012 微博情感分析评测报告”. In: (2012), pp. 1–8.

[4] 谢丽星. “基于 SVM 的中文微博情感分析的研究”. In: (2011), pp. 1–82.