Report for Final Project : Analysis of Rumor

Chen Ying, Zhu Simo, Liu Guoding, Mao Haining

January, 2019

Contents

---

1 Introduction

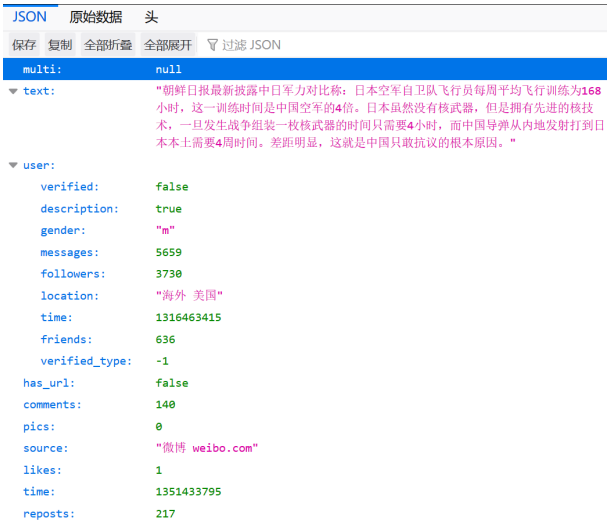
Text data is around us everywhere, and how to deal with those words, sentences or passages has long been a hot topic for natural language processing fields. With the rapid development and popularization of social media services, rumors are spreading with unprecedented rapidity and have a tremendous impact on human society. Meanwhile, the development of artificial intelligence technologies provides a promising approach for social media platform to automatically detect rumors.

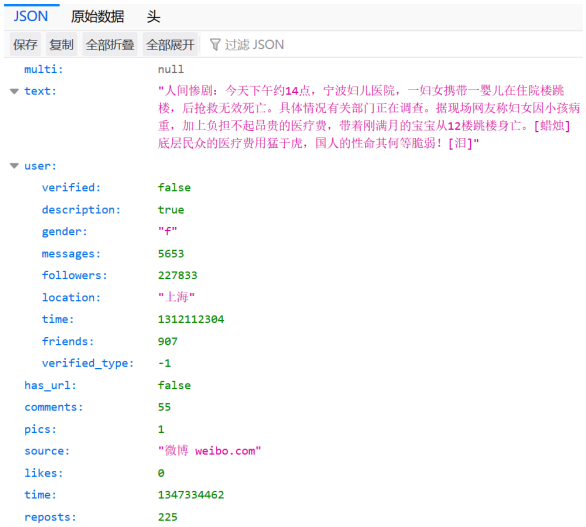
In daily life, people usually distinguish the authenticity of microblog events based on their common sense or through news websites and public communities, however, the reports of this kind of website media are incomplete and have certain time delay. Therefore, the automatic identification of rumor events can help us better prevent rumors and assist management agencies in rumor intervention and governance.

Under that situation, we hope to set up a model to distinguish those rumors from normal ones with only contents rather than making predictions based on all comments when users forward the message. In the following part, we will explain how the semantic and syntactic information influence the authenticity of a given context, how different representing methods and classification models performs on this classification problem, and, finally, how our primary features can be explanatory for deeper analysis of rumor.

2 Preliminary Process (Data Reading)

The documents in the dataset are scores of JSON (JavaScript Object Notation) documents, the information contained in one json document has the construct below:





The useful information of it contains:

- Text
- Weibo Features
  (a) Whether the weibo has URL
  (b) Number of comments
  (c) Pics
  (d) Sources of this weibo
  (e) Likes
  (f) The time when this weibo is sent
- User Features
  (a) Whether the user has description
  (b) Whether the user is verified and the verified type
  (c) The gender of user
  (d) Number of followers
  (e) Number of friends
  (f) The location of the user
  (g) The time when the user joined weibo
  (h) Number of messages user had sent

To get this information, we preprocess these json documents. In this process, we use the `'os'` & `'json'` module to read all the json data and extract information we need. Since the dataset has been processed, there is no missing data (in other words, all the json documents have complete information) or duplicate records, the presentation of features are also consistent, which saves us much effort.

Then we encode each feature by the method **ordinary encoding**. First, we use list to collect data, and then transform it into ndarray to adapt (except text data, which will be processed in the feature extraction part). In the meantime, we transform the data type into int (some examples are below).

```
1 has_url = np.array(has_url).astype(int)
2 verified = np.array(verified).astype(int)
3 description = np.array(description).astype(int)
4 gender = np.array(list(map(trans_gender, gender)))
5 followers = np.array(followers)
6 friends = np.array(friends)
7 category = np.array(category)
```

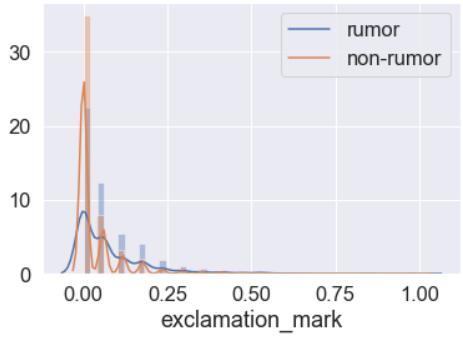
3 EDA & Feature Extraction

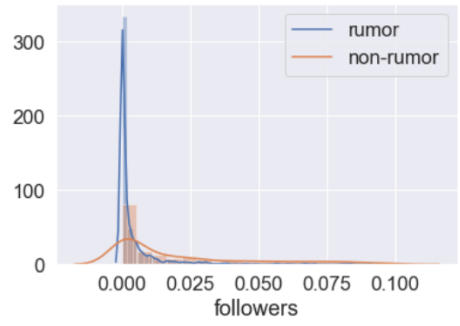
In this project, we try to extract features from the information in each json document. Before this project, we read some papers, among which the paper written by Prof. Liu Zhiyuan impressed us most. He asserts that there are several methods to detect rumors according to the features of the time series, but in that method, we can't detect rumors as soon as they are sent. Additionally, stopping the rumor early will help stop the potential social loss. This idea determines the features we finally choose. It should be attached great importance here that in this project we want to extract **the early characteristics of rumors**. The information of rumor after the time when rumor is sent, such as comments of this rumor isn't considered into this classification model.

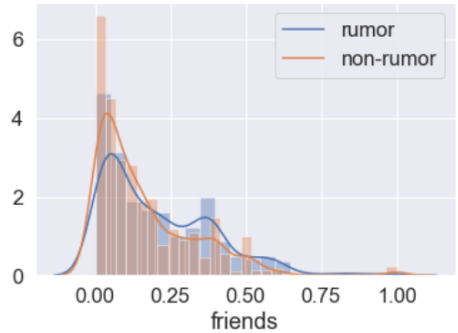
First we use Exploratory Data Analysis methods to see the general features of rumors in this dataset and then we select the features which will be used in the model training process. [1]

3.1 Exploratory Data Analysis (EDA)

In EDA process, we first explore the single feature of rumors as well as non-rumors and then compare them in one picture. For example we count the number of exclamation marks(**the coordinate has been normalized**) in each text and then describe it through frequency histogram to figure out the characteristic of distribution.



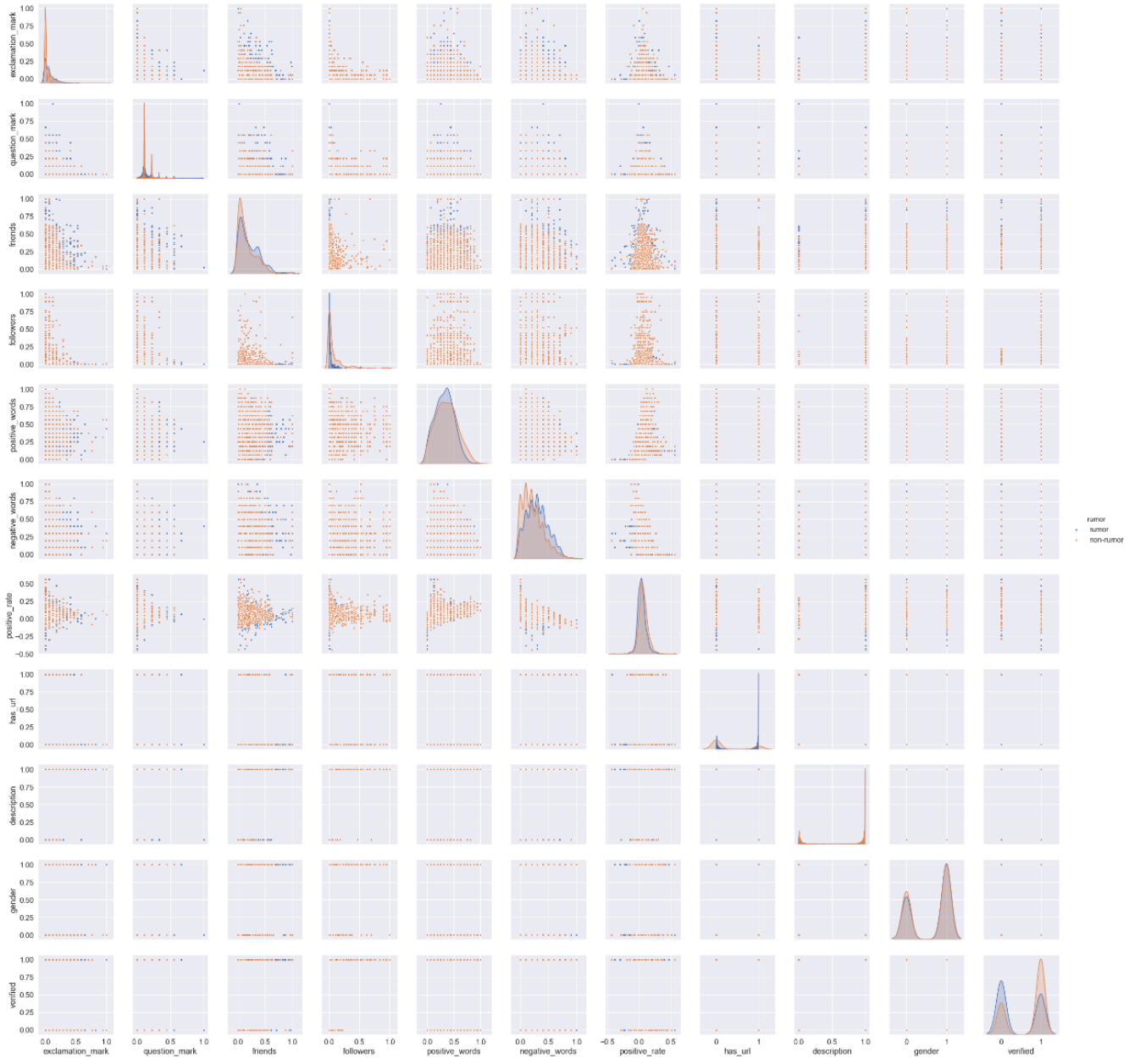




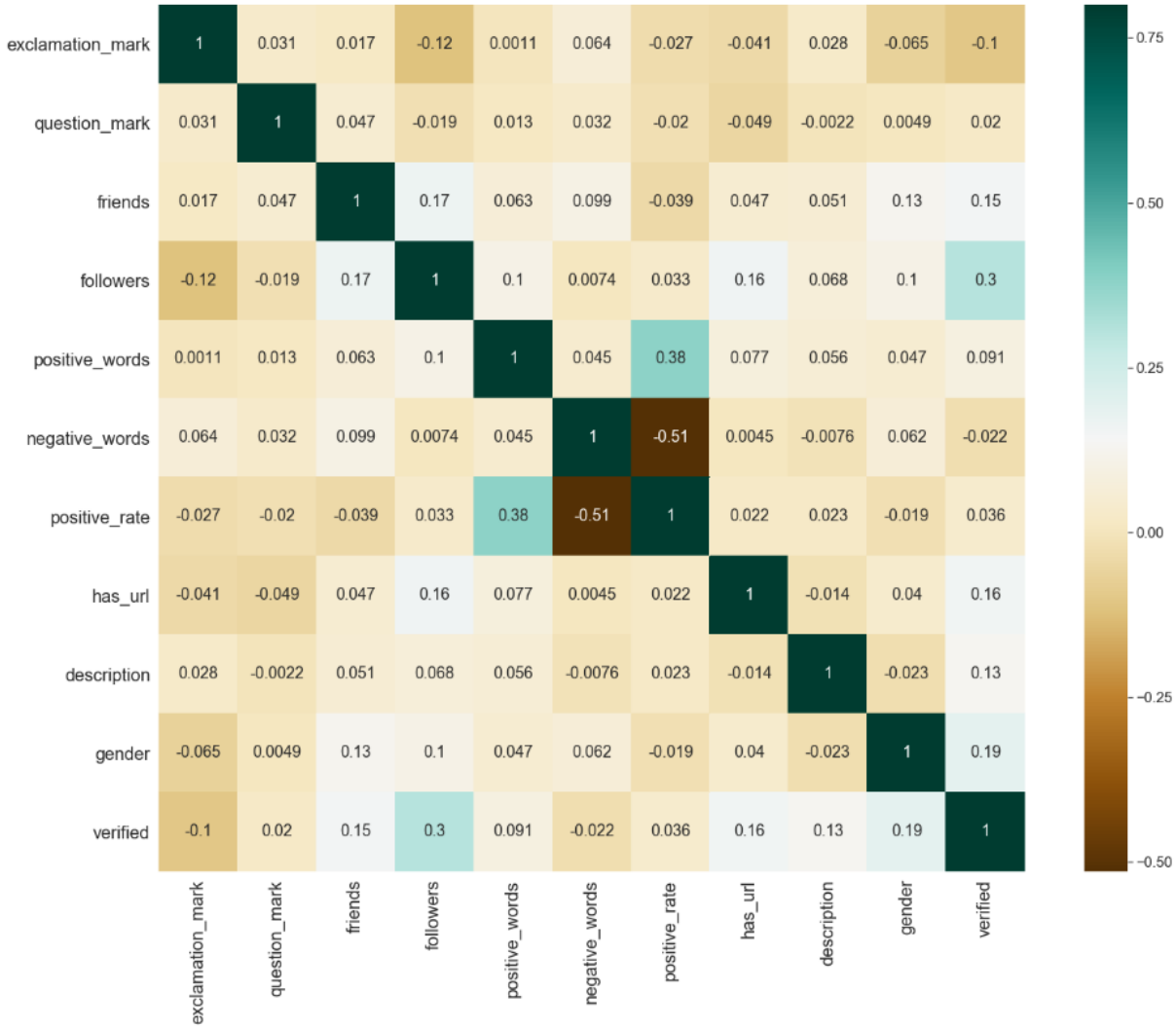
[1] In the real process, we must construct feature first and then use EDA to see the characteristics. After this we determine whether we will use these features. The order this report describe is little different from the real order.

In this process, we find that rumors are more frequently expressed with more exclamation marks as well as question marks. Besides, rumors tend to present negative attitude. Meanwhile, we find the number of friends of rumor-mongers' weibo account is statistically larger.

Then we explore **the correlation between different features** (variables). At the same time, we can see the features more clearly. Actually, none of these features are predominant feature which can almost determine the classification with a relatively high coefficient of association, so the combination of different features may be a good method.



We also find an interesting thing that the positive rate of the rumors is not always low, instead, it is very low or very high. That reminds us the rumors may be too positive or too negative, so decision tree may be a good choice. However, generally speaking, the rumors are always radical.



3.2 Feature Extraction

After the exploration process, considering the construct of the dataset and the characteristics of the rumor texts, we select the features below as the features we aim to extract and use as analytical basis in this classification model.

- Text Features
  - (a) Number of exclamation marks
  - (b) Number of question marks
  - (c) Number of positive words
  - (d) Number of negative words
  - (e) Positive rate of the text
- Weibo Features
  - (a) Whether the weibo has URL

- User Features
  (a) Whether the user has description
  (b) Whether the user is verified
  (c) The gender of user
  (d) Number of followers
  (e) Number of friends

Since the features (except text features) have been processed in the preliminary process, this time we only need to extract the features from weibo text message mainly through NLP (Natural Language Processing) methods.

This process is hard to handle. We refer to papers in the field of NLP and conclude that **rumors are always radical**. So we thought about those features which can reflect whether the text is "radical". According to the general knowledge of linguistics ~~(though quite poor)~~, finally we select these features to construct our rumor classification model:

- Number of exclamation marks
- Number of question marks
- Number of positive words
- Number of negative words
- Positive rate of the text

We use these features for the reason that if one text is radical, it will tend to use more exclamation marks and more question marks[2]. Besides, rumors tend to use more positive or negative words (radical, in summary). In general the tonal of the text will be radical so we try to construct a feature which will reflect this point.

We know the character of one sentence will be represented by **the "key word" of this sentence**. The "key word" in this project is defined:

> The words appeared in this sentence but don't appear in most of sentences we use in daily life.

So we want to get the "key word" and evaluate the text by the "key word" of the text. Then we use module jieba to split the raw text message (before this we use regular expression to clean the text) and get the most importance "key words" of the text (jieba has a corpus and can return the importance words according to the TF). The number of importance words can be set, which forms a hyper-parameter.

Once we get the "key words", we use SnowNLP module to evaluate the emotion of each word. The value of neutral word is 0.5. The value will be larger if the word is more positive and smaller if the word is more negative. So we count the positive words and negative words of these "key

[2]Just as the explanation we've proposed before in subsection 3.1

words" according to the rule: if the value is larger than 0.6, the word is positive word. If the value is smaller than 0.4, the word is negative word.

```
for x, w in jieba.analyse.textrank(sent, topK=top_word, withWeight=True,
    allowPOS=allowpos):
    rate += (SnowNLP(x).sentiments - 0.5) * w
    if (SnowNLP(x).sentiments < 0.4):
        negative_num += 1
    if (SnowNLP(x).sentiments > 0.6):
        postive_num += 1
##加上最后一个0.000000001是因为有的微博没有关键词，所以为了防止分母为0加上一个微小数
rate = rate / (len(jieba.analyse.textrank(sent, topK=top_word, withWeight=True
    , allowPOS=allowpos)) + 0.000000001)
```

At the same time, we calculate the positive rate of the whole sentence according to the "key words". The algorithm is:

- First get the evaluate value of each word and then subtract it by 0.5. Then get the average of this subtracted value. This value is the whole positive rate of the whole sentence.
- Meanwhile, we count the number of exclamation mark Number of question mark and then transform the data into array.
- Then we get the whole features we want to extract. All the features have been transformed into array and all the data types are int or double.

4 Model Selection and Model Training

4.1 Comparison between Several Models

In this model, we want to select a model that can classify whether a document is a rumor or not. Because we cannot easily tell the answer, we need a model that is highly interpretable. Here are some alternatives:

- KNeighborsClassifier
- Logistic regression
- Decision tree
- Random Forest
- Adaboost

The model of deep learning is so complex that we can hardly interpret it, so we didn't use neural networks.

Then we use cross-validation, split the dataset into training set (70%) and testing set (30%). Using the training-set to train the model and test it on the testing set, we can calculate the accuracy of the model. Initially we do not adjust the parameters of the model.

Initially we do not adjust the parameters of the model.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=50)
# change the shape of Y_train to (n_samples, ) using `.ravel()`
knn.fit(X_train, Y_train.ravel())
knn_pred = knn.predict(X_test)
print('The accuracy of the KNN is', metrics.accuracy_score(knn_pred, Y_test))
## The accuracy of the KNN is 0.7131313131313132
```

KNN uses the k nearest neighbors around a certain data to predict the classification of the data. As we can see, it doesn't perform satisfactorily.

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(penalty='l1',max_iter=800,fit_intercept=False,solver=
    'saga',C=20)
clf.fit(X_train, Y_train.ravel())
clf_pred = clf.predict(X_test)
print('The accuracy of the Logistic Regression is', metrics.accuracy_score(
    clf_pred, Y_test))
## The accuracy of the Logistic Regression is 0.7686868686868686
```

Logistic Regression uses a linear model to fit the sample points, and then uses a loss function to classify the data. In the process of adjusting the parameters of the LR model, we find that decreasing regular terms and setting the penalty to L1 significantly increases model accuracy. **This means we are dealing with a under-fitting problem**.

```
from sklearn import tree
dtree = tree.DecisionTreeClassifier(max_leaf_nodes=15, max_depth=30)
dtree = dtree.fit(X_train, Y_train.ravel())
dtree_pred = dtree.predict(X_test)
print('The accuracy of the Decision Tree is', metrics.accuracy_score(
    dtree_pred, Y_test))
## The accuracy of the Decision Tree is 0.7454545454545455
```

From the process of training Logistic Regression model, we can learn that the features are quite weak, thus not surprisingly the decision tree performs poorly.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(
n_estimators = 100,
criterion = 'gini',
random_state = 0
)
rfc.fit(X_train, Y_train.ravel())
```

```
rfc_pred = rfc.predict(X_test)
print('The accuracy of the Random Forest is', metrics.accuracy_score(rfc_pred,
    Y_test))
## The accuracy of the Random Forest is 0.7858585858585858
```

Due to **the requirement of interpretability**, we tried to use Random Forest to improve the performance of decision tree. Sample points are randomly selected from the sample pool each time to **reduce the negative impact of weak features**. RF performance is satisfactory. In this way, we can predict which microblogs may be rumors by calculating the importance of each feature, thereby reducing the cost of manual investigation.

```
from sklearn.ensemble import AdaBoostClassifier
abc = AdaBoostClassifier()
abc.fit(X_train, Y_train.ravel())
abc_pred = abc.predict(X_test)
print('The accuracy of the AdaBoost is', metrics.accuracy_score(abc_pred,
    Y_test))
## The accuracy of the AdaBoost is 0.7696969696969697
```

AdaBoost parallels weaker results and superimposes a stronger result to reduce bias. Due to weaker features, Ada also performs better than LR.

In summary, there' re multiple features, but all these features are not strong enough. Decision tree are simple to understand and interpret multiple-factor influence, but decision tree performs with unsatisfactory result.

Random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data. Random forest classifier corrects for decision trees' habit of overfitting to their training set.

4.2 Random Forest Model Training

Because the performance of Random Forest model is significantly better than other models, **we use Random Forest to train the classification model**. Then we adjust the parameters further, using GridSearchCV to get better parameters. After several rounds of GridSearchCV, accompanied by a reduction in range of hyperparameter space and a decrease in step length of GridSearch, we get the final model. In particular, we find that when we do not limit the max-depth and max-leaf-nodes, set estimated quantity to 118 and criterion to gini, we can get the highest accuracy.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(
n_estimators = 118,
criterion = 'gini',
random_state = 0
```

```
)
rfc.fit(X_train, Y_train.ravel())
rfc_pred = rfc.predict(X_test)
print('The accuracy of the Random Forest is', metrics.accuracy_score(rfc_pred,
    Y_test))
## The accuracy of the Random Forest is 0.7888888888888889
```

Limited by the quality of the original dataset, the accuracy of the model is 0.789.

RF can effectively reduce training errors. Diversity in individual tree prediction functions comes from bootstrap samples and randomized features for node splitting. And RF combines the diversity of prediction functions, **which effectively reduces the bias caused by weak features**. That's the reason why RF perform better in this problem.

Then we can calculate the importance of each feature with the model just trained to tell which feature we should care more.

```
rfcf=RandomForestClassifier(
n_estimators=118,
criterion = 'gini',
random_state = 0
)
rfcf.fit(X_train, Y_train.ravel())
for j in zip(feature_names, rfcf.feature_importances_):
print(j)
## ('exclamation_mark', 0.06132879347402999)
## ('question_mark', 0.023456691292703936)
## ('friends', 0.1518103983298208)
## ('followers', 0.3542632876955608)
## ('positive_words', 0.08046175973506683)
## ('negative_words', 0.07246633599533094)
## ('positive_rate', 0.15069153233769772)
## ('has_url', 0.026621527003040508)
## ('description', 0.008268131758053498)
## ('gender', 0.02351405594082627)
## ('verified', 0.04711748643786886)
## 其中依次对应：包含感叹号次数，包含问号次数，好友数，被关注量，正面词汇，负面词汇，
    正面率，是否包含链接，是否包含描述，性别，是否验证
```

We can find some interesting results from the importance of feature. As predicted previously, all features are weak. **The most important feature is the number of followers**. This is a **counter-intuitive** result that neither validation nor positive word rate matter. This may be because the dataset contains rumors of many low-level users who have few followers.

Another interesting finding is that although both positive and negative words are weak features, **the positive rate of a sentence is a relatively strong feature**. This could offer us some

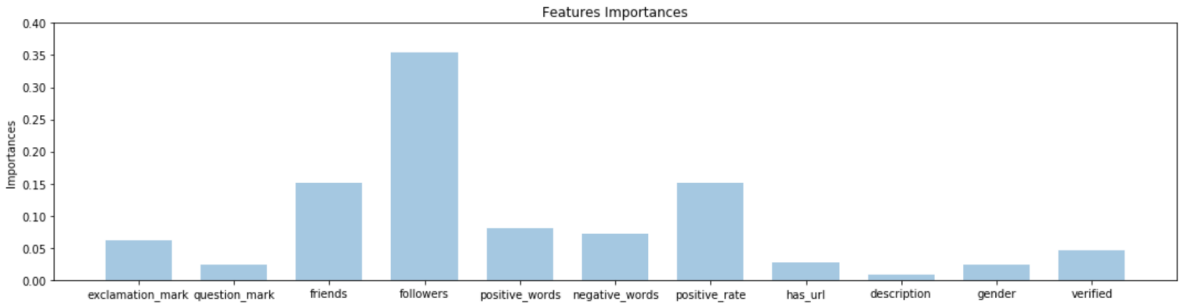
insights when identifying rumors.

5 Evaluation and Analysis

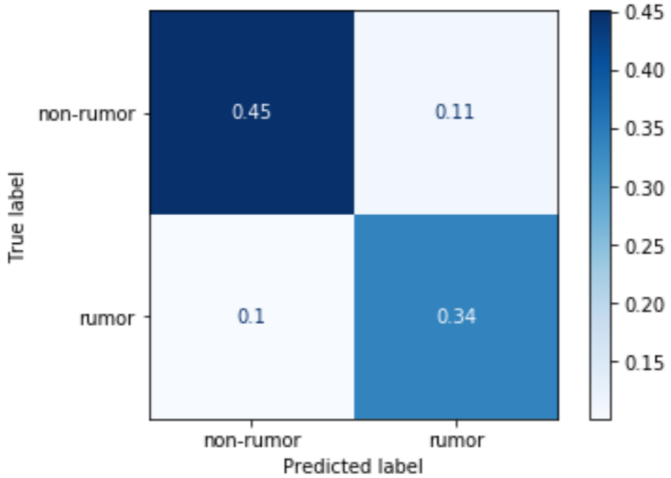
All contexts contain two kinds of information: **semantic information and syntactic information**. Study of the former can lead to judgement for each single word and phrase, and study of the latter focus more on the organization of texts. Above all, the key problem is the way to extract features from the text. In our project, we have found some interesting points.

5.1 Feature Importances & Confusion Matrix

The importance of each feature based on the Random Forest model has been shown in its entirety in subsection 4.2. Here we give a visualization graph to show the characteristic of feature importance distribution intuitively.



Additionally, to evaluate the accuracy and efficacy of the classification model, we calculated the confusion matrix as follows.



From a practical view, social media platform can neither ignore the propagation of rumors (which may undermine social stability and harmony) nor massively delete posts with an "unintentional model's accidental attack" (which may cause customer attrition). Therefore we take the

accuracy of both positive and negative classification into consideration, and we need to evaluate the efficacy of the model with the whole confusion matrix.

5.2 Analysis of the Result

We've already discussed the result in some detail, so I don't intend to reiterate those conclusions here, only enumerating several essentials.

- To some extent, the `'followers'` is a prominent factor to the classifier. (It' s quite counterintuitive, perhaps the generation method of the dataset causes bias. The phenomenon may be caused by survivor bias, or by normal generation mechanism, with unverified account with zero followers more likely to spread rumors around.)
- The tone of rumors tends to be extreme, and the `'positive_rate'` feature has a certain weight in rumor determination.
- None of these features plays a predominant role in decision making, so we need aggregation of features, which in turn proves the correctness of choosing Random Forest model.

6 Further Improvemets

The accuracy of our model is about 79%, partly due to the restriction of our small dataset, in that the generalization capacity may not support operating on enormous and complex dataset. We hope to improve the **efficacy of feature extraction by new model or proper hyperparameter** or **introduce early comments and reposts as new features** into the model. Also, we need a **larger and more representative dataset**.

In fact, due to other research, **the distribution of Internet rumors in various fields is extremely uneven**. For example, disclosed in the third quarter of 2018, of the 676 Internet rumors, **politics accounts for the largest proportion**, about 28%; news came in second with 22%; entertainment and business accounted for 13%, respectively; and 11%, third and fourth; as many as 18 other subcategories are not to 10%. **This imbalance makes rumor recognition possible in certain areas with high certainty, but the other area is not good, which also forms a challenge for rumor detection project.**

Domain information with different data distributions is often treated as a whole, thus the classifier established in this way often over-sensitive. As a result, Internet rumors in some areas are not well identified. We need to build **separate classifiers for each domain to capture specific fields**, otherwise, high-frequency domain words will be misleading.

In the future work, on the one hand, we will deeply **investigate and analyze** the influence of the features mentioned above on the detection effect of rumor events, so as to select the best combination of features. On the other hand, we will consider **the principle of event propagation** and construct a **temporal feature representation model** that is more consistent with event evolution and propagation. At the same time, we will also consider using other model to solve the problems of complex **artificial feature construction and weak feature semantics**.