CS579 Final project Report (Still writing)

Jiada Tu A20306906
Lv Zhang

1. Introduction

2. Data

When we get a tweet from the twitter API, the tweet contains a field names “retweet_count”. This means how many people click the “retweet” button under the tweet. We call this “system retweet”. But many people likes to “manually retweet” the tweet, which means that they retweet a tweet by post a new tweet: typing their comment/reply and add “RT @xxx [original tweet content that be retweeted]” behind. This kind of “retweet” is not recorded by twitter system and it's hard for us to collect the “manual retweets” of a tweet. So, we just focus on the training and predicting “system retweet” with our model.

We didn't trace a certain tweet and record it's retweet-count changing as time goes by. Instead, we collect discrete data points which contain tuple (“time passed”, tweet) in our data set.

Because the retweet count of a tweet increase as time goes by, retweet count is related to when we collect that tweet. For each single tweet that we collect, we record the timestamp when we get that tweet. So, through “(the time we collect the tweet) – (the time the tweet is posted)”, we can get the “time passed” information. Tuple (“time passed”, tweet) become a data point in our dataset.

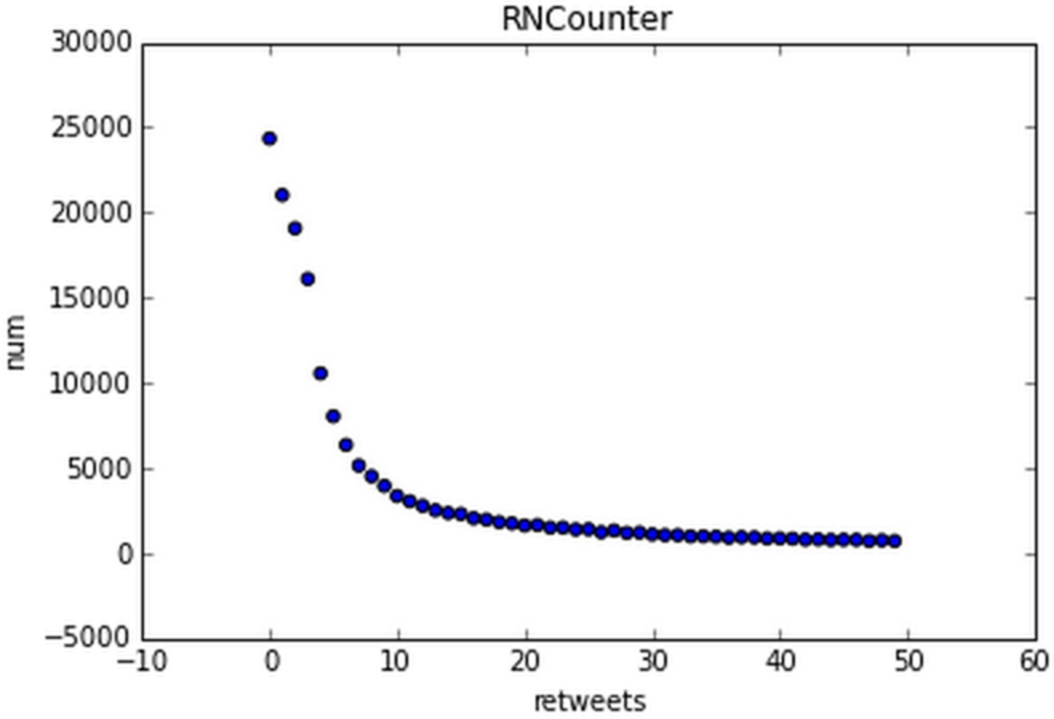
We only collect the tweets that's in english, because some languages are hard to tokenize and it eases the rest work.

We collect data using both Stream API and REST API, because each of them has some benefits and they can make up each other.

For Stream API, it doesn't have the 15 min limitation. But it only returns tweets which are in the most recent few seconds. So, the data we collected by this API will never contains data points that has 0 retweet count and “time passed > 30 sec”, because its original tweet is the only chance that can be shown by Stream API.

REST API don't have this problem. We use Stream API to get user id from the recent sample tweet and get their user_timeline(600 at most). But it's slow base on the 15 minutes limitation.

One problem of the data we collected is that it's highly imbalanced. More than ?% of all tweets in REST API has 0 retweet count, and more than ?% of the rest have 1 retweet count. This **may** make our model learned wrong parameter (Especially linear regression. Because as times passed, many tweet still gets 0 retweet. Then the linear regression will think “time passed” is not related to the retweet count at all, when in fact, it is). So we try to decrease the ratio of the date points that has 0,1,2 retweet counts, but not so much that makes our data set totally unrelated to the real ratio. Then we get the below retweet-count distribution:



The x axis is the retweet count and the y axis is the number of the tweets that has this certain retweet count.

We mix the data collected by these two API. After weighting, balancing and sampling(the data set is too large to fit in memory when do the training and predict) the data, we get 350K tweets.

One more thing. Actually we classify the tweets into 24 different files base on which hour (UTC time) in a day that the tweet is posted. We do this in the assumption that the retweet pattern of at tweet is different if it's posted in midnight from if it's posted at afternoon. So then we can train the model separately with these 24 data sets. But the tweet is global, so this assumption maybe wrong. Until now, we haven't do the training and predicting base on this assumption, what we do is just using all 24 datasets. We put this into future work to figure out if this assumption can improve the accuracy of our model.

3. Methods

Because it's hard to predict the exact number of a tweet, we train and predict the log2(retweet num). Then if our predict is 2 different from the real num, then it's within [¼ x, 4x] range, when x=the real retweet num.

We figure out some attributes that may related to the retweet count. Mainly: the followers of the user, the friends of the user, the time passed by, the list that user is a member of, the number of status that user have (total number of tweets), the sentiment of the tweet / user description, if the tweet has url, hashtag and image, etc.. The number of some attributes can be very large (like time-passed, followers number, etc.) and we think it will not make much difference if it's “very large” or “very very large”, so we do log2() of them.

For the sentiment analysis, we use word list with sentiment score. For each tweet / user description, we tokenize them, match the tokens to the word list and add the sentiment scores (separate positive and negative score). These scores became a part of the attributes of the dataset (X matrix).

For the model, we do both regression (mainly) and classification. To evaluate the regression, we use Pearson correlation coefficient, mean squared error and relative squared error. To evaluate the classification, we use the accuracy score.

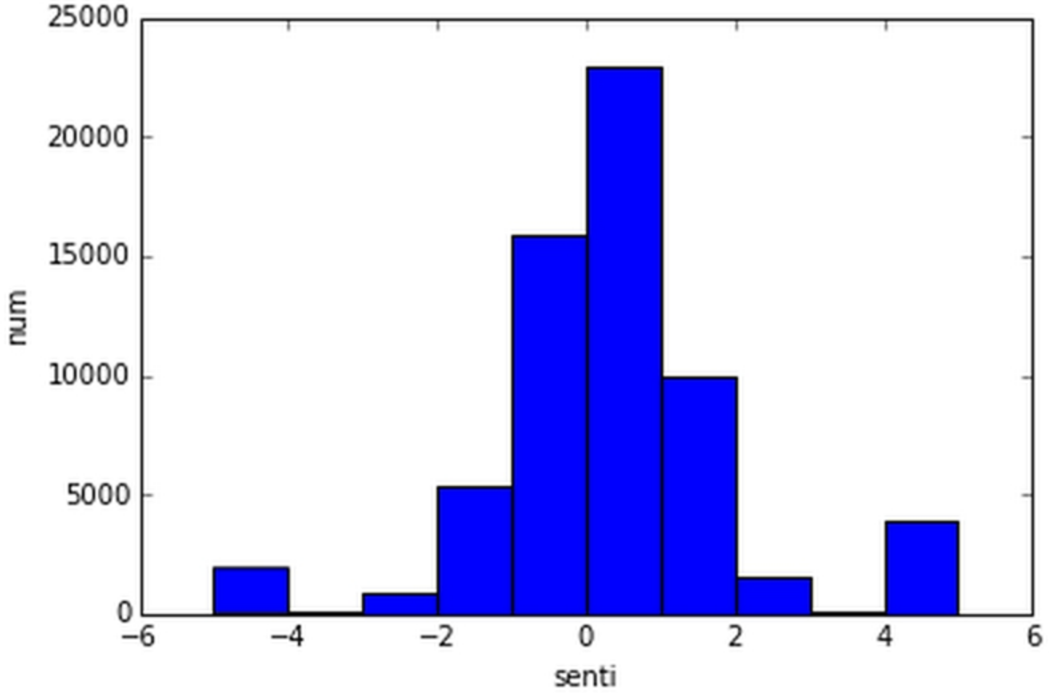
For regression, we use linear regression at first. Then we figure out that most of the parameter is not linearity with the retweet count. For example, assume y=retweet count, x1 and x2 are some parameters, then y may not equals “x1+x2” but “log(sqrt(x1))pow(x1,2)+x2”. Also, y may not in relationship with them as y=ax1+bx2, but y=cx2*x2 or y=d x1^x2. So in these situations linear regression is not good and we need some method that's not linear. Then we find decision tree. To refine the result of decision tree, we mix bagging method with it. So it becomes the random forest. What it basically do is has many decision tree, each of them are trained on subset of the dataset and attributes. Then each of them will do the predict and the final prediction is the weighted result of predictions by all of them.

For classification, we also use decision tree with bagging.

4. Experiments

A. Sentiment analysis:

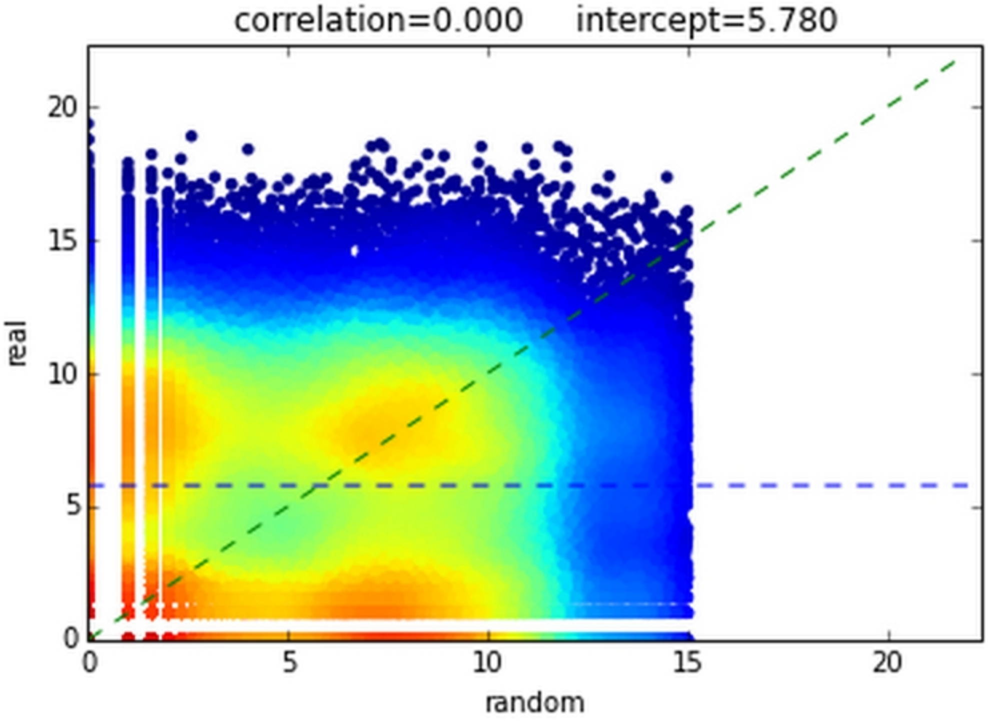
We use the word list from [1]. It is based on the Sentiment 140 [2], a twitter sentiment analysis product. It's in the same form as AFINN word list, but it has much more terms: totally 62,468 terms. And here's the sentiment score distribution of the terms in it:



The x axis is the sentiment score, and the y axis is the number of words that has this certain sentiment score.

B. Baseline for regression:

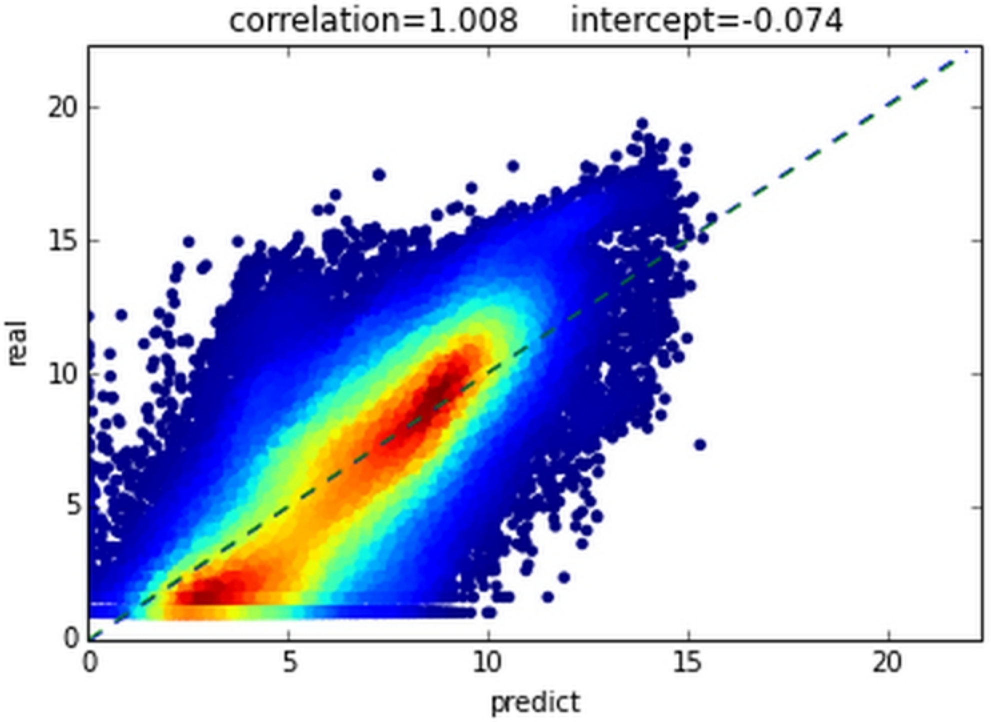
To establish a baseline, we make a naïve predict model that randomly predict the retweet count base on the distribution of the retweet-count of the dataset. So if 20% of the tweets has 0 retweet, then the model will have 20% chance to predict 0 for any tweet. Here's the density chart of the baseline model:



The x axis is the predict log2(retweet_count), the y axis is the real log2(retweet_count). The green line is just a diagonal y=x. The blue line is a line (y=ax+b) that try to fit the points in the figure, and the “correlation” is 'a' of the blue line (y=ax+b), “intercept” is 'b' of that. **The Pearson score is (0.000156, 0.967), the mean squared error is 5.85, and the relative squared error is -0.986.**

C. Linear regression

We train on 80% of the dataset and predict on 20% of the dataset. I didn't do the cross validation because the score is basically the same as when I train the model on 20% and predict on 3% of the dataset (for test), so I believe no cross validation is need now. But we may add that in the future. We get this result:



The Pearson score is (0.730, 0.0), the mse is 2.56, and the rse is 0.533.

We can see the blue line is basically match with the green line, which means the bias is small, maybe. And the total predict trend to be much better than the random predict model. But the tilt of the red area seems to be not match with the y=x. We will talk about that later, now let's see the coefficient of the attributes that the model learns:

```
coef=
('log_user_followers_count', 0.65295531605579116)
('if_images', 0.54330618226449623)
('log_time_passed', 0.2181920959454299)
('sentiment_text_positive', 0.20258102517539073)
('user_geo_enabled', 0.088063682906177096)
('sentiment_description_negative', 0.086829846933841881)
('log_user_friends_count', 0.014099557460164783)
('user_favourites_count', 5.2903256567244223e-06)
('user_protected', 0.0)
('sentiment_description_ratio', -0.044158707916442774)
('sentiment_description_positive', -0.055423385617983059)
('user_verified', -0.075132856370884529)
('sentiment_text_negative', -0.101555590752608)
('log_user_listed_count', -0.12057422214411126)
('sentiment_text_ratio', -0.18724888042552093)
('if_hashtags', -0.23898136302832043)
('log_user_statuses_count', -0.26923089865881944)
('if_mentions', -0.61312403567229634)
intercept= -1.94397178177
```

We can see, the log(follower count) matters the most. Basically, if you have 8 times followers as now you have now, all your tweets trends to be have 4 times retweet-count as now. Also, if your tweet has images in it, it will more likely to be retweet. The time passed should be some kind of significant and it is, so it's good. Also, if sentiment_text_positive shows that if you

5. Related work

[1]NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets, by Mohammad, Saif and Kiritchenko, Svetlana and Zhu, Xiaodan, June, 2013
[2]http://www.sentiment140.com/

6. Conclusions and Future work

1) Train the model base separately base on the UTC time hour it is post and see if this affect the accuracy.
2) Separate the tweets into few parts base on the time passed ([0, 10 min], (10 min, 1 hour], (1 hour, 1day], etc.) and fit the model on them separately.
3) Try other machine learning methods which may fit better with the attributes (for example: not (time_passed+follower_num) but (time_passed*follower_num)) to do the prediction.

7. Contribution