Best tweet post date prediction

Individual interdisciplinary project 2017/2018
Supervisors: Tobias Scheffer & Paul Prasse
Submission Date: 02.03.2018

Victor Lombardi

University of Potsdam, Germany

Abstract. Social media platforms such as Twitter encourage people to react on publications in order to spread information. On Twitter, retweeting is the most important information propagation mechanism. This paper investigates the problem of predicting the popularity of messages as measured by the number of future retweets and favorites (non-famous users restricted). The solution starts by crawling a large data set, then different models are trained, tuned, and evaluated. For the purpose to access the prediction easily, a web-based prototype was made.

Keywords: Twitter, machine learning, crawler, data science, social media, data set

1 Introduction

With 200 million tweets per day in 2011[1], Twitter has remained for more than 10 years one of the largest social networks. Its success is due to the simplicity of the Tweet concept, but also because Twitter has grown from a social media platform to an increasingly important news medium (CNN, BBC, etc.) and marketing tool that enables brand promotion and direct announcements to clients (Movies, Products, etc.). With Twitter, businesses can grow by reaching larger audiences, due to a phenomenon called "Viral Marketing" (via Retweet (RT), which allows a user to re-post a tweet on their wall, publishing it to their followers).

There are many tools on the web that give the user analytics on their tweets. Users can see graphs of representing changes in the size of their audience over time, but the most of the analytics do not take into account the content of tweets.

In order to create a model that is able to predict a score based on millions of tweets, a data set has to be built first, which is pointed out in section 2. Section 3 presents the approach supported. In part 3.1, features and outputs are exposed. Then baselines will be introduced. In part 3.3, the two models picked out will be presented and tuned. For this purpose, data sets of different size are used. Then in section 3.4, one will be selected, evaluated and improved. Lastly, section 4 presents the web client made in order to give access for a usual user to predictions.

[1] `https://www.dsayce.com/social-media/tweets-day/`

2 Mining a dataset

The first step was to write my own crawler that retrieves the latest available 3,200 posts for each user. This crawler is available on GitHub[2] and its usage is documented inside the repository.

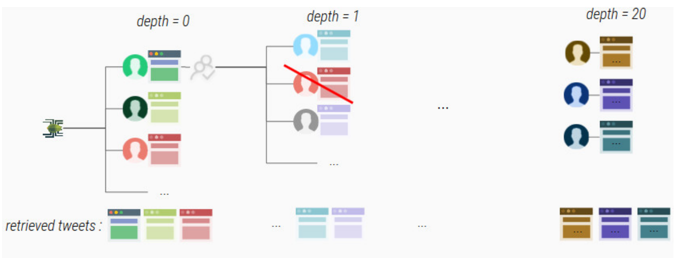


Fig. 1. Crawler principle

As illustrated in figure 1, the script identifies a starting point using a set of random users based on my own followers.
From this point, a set of 200 followers is requested for each user, duplicates are removed and the algorithm determines recursively new users.
After executing the script with a maximum depth of 20 or a maximum number of tweets to retrieve specified, a set of more than 400 random accounts was picked out.
Using this set of users more than 585,000 tweets, including relevant data for processing (number of RT, user info such as number of followers, etc), were retrieved.

This crawling method is quite powerful because by alternating requests between users' time lines and users' followers, the requests limitation imposed by twitter is less easily reached than if we were acting brutally. In addition, with a depth of 20, the potential maximum of tweets reachable is about 10^{115}.

3 Modelization

In this section, I elaborate how to predict the popularity of tweets in detail. The first step is to select some relevant features to create input. For future evaluation, it's important to build baselines that will be described

[2] `https://github.com/vykimo/twitter_best_date`

hereafter. Then different models are introduced with their tuning.
And finally, I'll present feature importance in model process.

3.1 Feature selection

Table 1. Input

Text	Word2Vec score by words in sentence
Hashtag	Word2Vec score by letters in word
Weekday	From 0 (Sunday) to 6 (Saturday)
Hour	Tick in seconds
Followers_count	Number of followers
Friends_count	Number of friends
Listed_count	Number of lists
Statuses_count	Total number of tweets
Link	Number of http links inside text
Quote, . . . , !, ?	Counting of the punctuation
@	Number of mentions
Upper	Number of uppercase letters
Polarity	Float within the range [-1.0, 1.0]
Subjectivity	Range from 0.0 (very objective) to 1.0 (very subjective)

As seen in table 1, inputs are mainly text-based (green inputs). Yellow inputs are account bound and reds are temporal information.
I built 2 Word2Vec models: the first one is trained with words in sentences and the second one is related to letters in a hashtag. This idea was pushed forward because hashtags are most often acronyms or small words.

In addition, I used TextBlob which is a Python library for processing textual data. I focused on the sentiment property, which returns a named tuple of the form Sentiment (polarity, subjectivity).
The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

Table 2. Output

Score	$NumberofRT \times 2 + NumberofFAV$
Ratio score	$\frac{NumberofRT \times 100}{Numberoffollowers}$

The output figured on table 2 is divided into two scores. The first one is a simple total of number of interactions (FAV and RT) with a factor 2 in order to represent that Retweet are generally much more important in the process of Tweet spreading.
The second output was created later in order to represent the popularity of the tweet inside the user's community (a ratio score greater than 100 represents a tweet that surpasses your usual follower audience). It will be a very useful score for evaluation.

3.2 Baselines

Four baselines based on hashtags were set in order to compare model performances:

1. Overall mean score for the whole dataset
2. Maximum mean for hashtags in a tweet : we calculate hashtag mean score in the entire database and we compare with other hashtags present in the tweet.
3. Mean for user: same as the 2nd baseline but restricted to user's timeline.
4. Maximum mean for hashtag for user only in user's timeline.

3.3 Models

Two different models were trained separately in order to solve this problem:

- Support Vector Regression (SVR)
- Random Forest Regression (RFR)

The SVR uses the same principles as the Support Vector Machine (SVM) for classification, with only a few minor differences.

Models Tuning

The SVR was trained with RBF kernel using scikit-learn[1]. In this case, I had to tune two hyper-parameters: $1 < C < 100$ and $0 < gamma < 50$. I did a nested cross-validation with a budget of 150 function evaluations. Each tuple of hyper-parameters was evaluated using twice-iterated 5-fold cross-validation.
Here are results:

- Support Vector Regression
  - Kernel = RBF
  - C = 99.95588
  - gamma = 22.38053

Regarding the Random Forest Regression, two tunings were made.
The first step was to do a random search to determine the range for each hyper-parameter. After that, I listed every combination of settings to try. I did this with "GridSearchCV", a method that, instead of sampling randomly from a distribution, evaluates all combinations we define. To use Grid Search, I set this parameter grid:

- 'bootstrap': [True, False],
- 'max_depth': [80, 90, 100, 110],
- 'max_features': [2, 3],
- 'min_samples_split': [8, 10, 12],
- 'n_estimators': [100, 200, 300, 1000]

This contains 960 combinations of settings to try out.
Here is the best combination found:

– Random Forest Regression
  - bootstrap = True
  - max_depth = None
  - max_features = auto
  - min_samples_split = 2
  - n_estimators = 15
  - n_jobs = -1

The second tuning focused on the hyper-parameter "sample_leaf_options". I simply calculated the Mean Square Error (MSE) for different models with distinct parameters and compared it to the lowest MSE of the four baselines.

Table 3. Baselines for different data set size + tuning on min_sample_leaf

Dataset Size	min_samples _leaf	Model MSE	Lowest Baseline MSE
22963	1	$3.71 \cdot e^{08}$	$3.87 \cdot e^{08}$
	5	$3.65 \cdot e^{08}$	$3.87 \cdot e^{08}$
	10	$3.62 \cdot e^{08}$	$3.87 \cdot e^{08}$
	50	$3.65 \cdot e^{08}$	$3.87 \cdot e^{08}$
	100	$3.68 \cdot e^{08}$	$3.87 \cdot e^{08}$
	200	$3.77 \cdot e^{08}$	$3.87 \cdot e^{08}$
	500	$3.78 \cdot e^{08}$	$3.87 \cdot e^{08}$
584753	1	$3.39 \cdot e^{08}$	$4.22 \cdot e^{08}$
	5	$3.51 \cdot e^{08}$	$4.22 \cdot e^{08}$
	10	$3.69 \cdot e^{08}$	$4.22 \cdot e^{08}$
	50	$3.98 \cdot e^{08}$	$4.22 \cdot e^{08}$
	100	$4.05 \cdot e^{08}$	$4.22 \cdot e^{08}$
	200	$4.06 \cdot e^{08}$	$4.22 \cdot e^{08}$
	500	$4.10 \cdot e^{08}$	$4.22 \cdot e^{08}$

As you can see on table 3, for the largest dataset, the lowest MSE is for min_samples_leaf = 1.

3.4 Evaluation

To evaluate models, I separated the shuffled dataset into a train set $^3/_4$ and a test set $^1/_4$. The first observation was that RFR is way faster than SVR [3].

Table 4. MSE for different models and data sets

Dataset Size	Model Type	Model MSE
22963	RFR	$3.71 \cdot e^{08}$
	SVR	$3.63 \cdot e^{08}$
584753	**RFR**	$3.39 \cdot e^{08}$
	SVR	$7.33 \cdot e^{08}$

MSE was printed for two different data sets and, as seen on table 4, for the largest data set, Random Forest Regression has the lowest error. And compared to the lowest baseline MSE on table 3, this model MSE is quite better.

[3] based on standard i7-6700K CPU @ 4.0 GHz

Therefore Random Forest Regression was selected for evaluation and improvements.

Improvements

The next step is now to try to reduce the MSE, which is really high. This is mainly due to the fact that some values, such as "Buzz" Tweets, are considered as aberrant values by the data model.
Therefore, I removed all the tweets exceeding a 300% performance ratio score from the data set (see table 2).

Similarly, the goal of the project is to predict a score for a "random" user, not for a popular star.
Thus tweets from user with more than 10,000 followers and tweets with an unreasonable score ($> 20,000$) were removed from the data set.
With these improvements, MSE was significantly reduced to $3.12 \cdot e^{06}$ (see real by predicted values graph on figure 2) and the R^2 of the model increased to :

$$R^2 = \texttt{[0.63 \quad 0.55]}$$

(respectively `[score ratio_score]`)

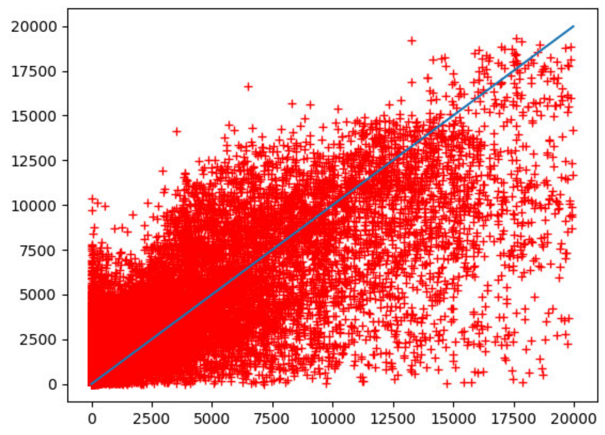


Fig. 2. Real by predicted values

Score classification

In order to show results in a more understandable way, tweets were classified into 3 classes based on ratio score from output (see table 2):

1. Unpopular tweet (ratio score between 0 and 49)
2. Normal tweet (ratio score between 50 and 199)
3. Very popular tweet (ratio score between 200 and 300)

A previous step to do before calculating metrics was to balance the amount of tweets for each class.
Once done, I calculated precision, recall and f1-score of the predictions on the test set. And finally I obtained rather satisfying results as reported on table 5.

Table 5. Classification report

	precision	recall	f1-score
class 1	0.88	0.77	0.82
class 2	0.68	0.89	0.77
class 3	0.92	0.46	0.62
avg / total	0.80	0.77	0.77

3.5 Feature importance

For the Random Forest Regression (RFR), it is possible to print the importance of each input feature.

Table 6. Feature Importance

Feature	Importance
text	0,5484
hour	0,1389
upper	0,0595
polarity	0,0437
followers_count	0,0356
subjectivity	0,0315
weekday	0,0313
statuses_count	0,0292
hashtag	0,0236
listed_count	0,0212
friends_count	0,0153
!	0,0096
@	0,0064
?	0,0058
quote	0
link	0
...	0

As supposed, the text on table 6 is at the first ranked with more than 54% of importance. In a surprising way, punctuation is irrelevant (including link and @ which correspond to a mention).

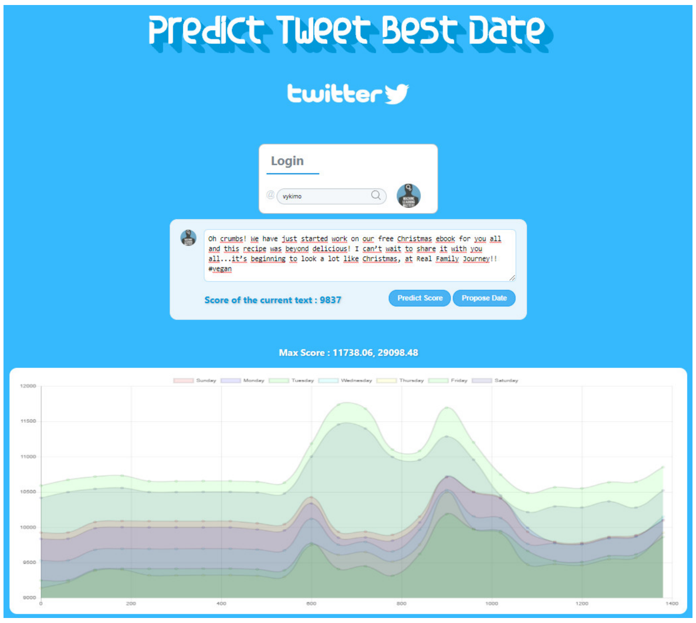
4 Web client

In order to access the predictions, I built a python-based web server with a fancy UI (figure 3). This server is available on GitHub[4] and its usage is documented inside the repository.

The first step consists in linking your twitter account (or someone else's) by simply entering the twitter user name (used for @mention). This step is made in order to retrieve account bound information used for features in input.

Once done, you can write the tweet you want to post in a text area. By clicking on "Predict Score", the score of your tweet appears with an indicator of the possible

[4] https://github.com/vykimo/twitter_best_date



Oh crumbs! We have just started work on our free Christmas ebook for you all and this recipe was beyond delicious! I can't wait to share it with you all...it's beginning to look a lot like Christmas, at Real Family Journey!! #vegan

Score of the current text : 9837

Max Score : 11738.06, 29098.48

Fig. 3. Web Client

popularity of the tweet if it was posted just now (divided between 'not popular', 'normal', and 'very popular'). You can play with the tweet content in order to maximize the score.
In addition, if you click on "Propose Date", after a small loading time, a calendar will be displayed. It shows popularity score evolution by hour and for each week day.
According to this prediction, you can have an idea of when posting this tweet could be the most interesting, in order to reach a maximum of people.

5 Conclusion

To conclude, the approach presented in this paper is very simple but the topic is actually much more complex than I first expected; its range is very wide and it needs further research. However I still got correct results and they seem consistent with reality.

For the moment, Random Forest Regressor gives better results for a larger data set and is way faster than SVR, which is only good for small data sets.

Further improvements may involve training the model on a bigger data set, to train other model types such as time series, and to merge them. Another important point is that currently I have no previous state of accounts, so currently retrieving old tweets is not very relevant because the number of followers, friends, etc. is taken at its last state. A particular intent is needed regarding this point.
In the same way, "Buzz" tweets, that have been put aside in part 3.4, could be treated by another specific model in order to include this very important principle which is characteristic of social networks.

Acronyms

FAV Favorite. 1, 2

MSE Mean Square Error. 3

RBF Radial Basis Function. 2
RFR Random Forest Regression. 2, 3
RT Retweet. 1, 2

SVM Support Vector Machine. 2
SVR Support Vector Regression. 2, 3

References

1. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
2. W. Koehrsen, "Hyperparameter tuning the random forest in python," *Towards Data Science*, 2018.
3. J. S. Marc Claesen and D. Popovic, "sklearn: Svm regression," 2014.
4. A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Citeseer, 1998.
5. Q. Y. Yang Zhang, Zhiheng Xu, "Predicting popularity of messages in twitter using a feature-weighted model," in *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences Beijing, China, 100190*, p. 7, 2012.
6. H. S. Bo Wu, "Analyzing and predicting news popularity on twitter," in *Department of Electrical and Computer Engineering, Clemson University, United States*, p. 10, 2015.
7. A. W. Sean Gransee, Ryan McAfee, "Twitter retweet prediction," in *Northwestern University, United States*, p. 2, Machine Learning, EECS 349, Bryan Pardo, 2015.