Blowing up the Twittersphere: Predicting the Optimal Time to Tweet

Zach Ellison (zellison@stanford.edu)
Undergraduate BS Computer Science Candidate,
Stanford University

Seth Hildick-Smith (sethjhs@stanford.edu)
Undergraduate BS Computer Science Candidate,
Stanford University

Abstract

As social media plays a growing role in society, more and more people are depending on social media for outreach and advertising. As this dependence grows, so does the importance of understanding how to make posts as effective as possible. In this project, we explore some of the factors that make posts effective, create a predictor to determine how successful posts will be, and then use that predictor in order to determine how to optimize one of those factors, time. Specifically, we examine tweets on Twitter and attempt to give the time that will optimize the effectiveness of a tweet with given text and by a specific user.

Keywords: Machine Learning; Regression; Twitter; tweet

Introduction

We can separate our problem into a few different steps. First, we need to model information about a tweet and how successful a given tweet is. Second, given a tweet, user, and post time, we must predict how successful that tweet will be. Finally, we then need to use our predictor to determine the optimal time for a given user to post a specific tweet, i.e. what time maximizes our success prediction for a specific user and tweet.

We considered two papers that address similar problems of using Machine Learning to understand interactions in social media and predict success of online content. Lakkaruja, McAuley, and Leskovec consider the connections between title, content and community in social media. From their work, we saw the benefits of breaking features into different models in order to better understand which types of features were having the greatest impact on our final predictions. They also did reasonably extensive language modeling when considering titles, which we considered when designing our own language model to examine the effect of the text of the tweets. (H. Lakkaraju & Leskovec,2013) Tsagkias, Weerkamp, and de Rijke wrote a paper considering a similar problem to ours as they attempted to predict the number of comments an online news article would receive. We were able to draw from their technique of first classifying an article to determine if it would receive any comments before then running a regression to determine how many comments it would receive when we were handling issues of sparsity in our data set. (M. Tsagkias & Rijke,2009)

Data

Our data comes from scraping tweets using the Twitter Search API. We utilize the tweepy.py Python package[1] to connect and query the Twitter API (we utilized portions of scripts from an assignment in MS&E 331[2], taught by Sharad Goel). We work with two datasets, 19,784 original tweets (i.e. not themselves retweets) that all contain the word stanford (our 'Stanford' dataset) and 243,706 original tweets that all contain the word 'california' (our 'California' dataset). Both datasets comes from the six day period of November 7, 2014 to November 13, 2014 and are split into a testing set of $\frac{1}{10}^{th}$ the total dataset and a training set of the remaining tweets. We use the testing-training split as a preliminary test of generalizability and ultimately apply k-fold cross validation to test the generalization error of our algorithms. Our dataset is notably sparse: only 10.63% of the 263,490 total tweets collected achieved a non-zero number of retweets.

From the very outset we were able to identify several interesting patterns in the data by simply plotting the average retweets per tweet and the total traffic over the day (see Figure 1). The figure demonstrates that retweet rates are not driven by traffic, in fact we observe that a number of points through the day there is an inverse relationship between the retweet rate and the number of tweets posted. The plot additionally demonstrates through filtering by retweet success that the shape of the retweet rate curve is not a result of large retweet outliers.

Features

There are a few different components we need to take into account when working with Twitter. Because Twitter is a social media site, we need to take into account the user themselves and their relationship to the Twitter community. In order to measure the efficacy of tweets, we also need to consider the quality of the content of the tweet. Finally, we need to consider the time that the tweet was posted to take into account the value of a given time. We therefore have three different feature categories that utilize different raw inputs to address these factors. We have User Features to account for a user's relationship to the Twitter Community, Language Features to account for the quality of the tweets, and a Time Feature to account for the activity of the Twitter Community at a given

[1] Roesslein
[2] Goel

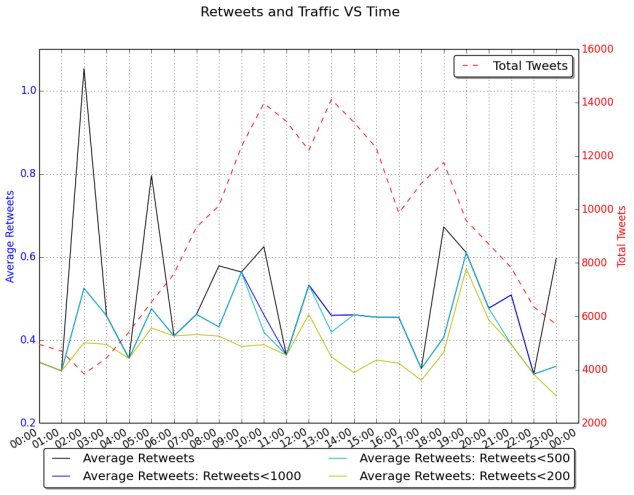

Figure 1: Retweet Rates And Traffic Over a Day

time.

Table 1: Raw Input

Feature	Description	Type
	User Input	
followers_count	Number of followers a user has	int
friends_count	Number of friends a user has	int
listed_count	Number of lists a user is on	int
favourites_count	Number of times a user is favorited	int
statuses_count	Number of statuses that user has posted	int
	Language Input	
text	The text of the tweet	string
	Time Input	
created_at	Unix timestamp the tweet was created at	int

Raw Input

The raw user data includes: $followers\_count$, $friends\_count$, $listed\_count$, $favourites\_count$, $statuses\_count$, $text$ and $created\_at$ (see Table 1). We take this data from the json formatted tweets in our dataset.

We use statistics about a users popularity to and the amount of interaction they have with Twitter to account for their relationship with the Twitter community. We would expect that users that are more well liked or more popular (higher followers, more friends, higher listed and favourites counts) to generally have more successful tweets because they have a larger number of people watching for their tweets. Similarly, we expect users with a high number of posted statuses to have more retweets because they receive more exposure within the Twitter community by virtue of having more information out there.

The time feature includes only: *created_at*. This model is relatively simple. It keeps track of the time at which a given tweet was created. We convert all times to PST from UTC. In this conversion step we assume all tweets in our Stanford and California datasets are posted in, or directed at California. This assumption does not add noise to the dataset as all times

Table 2: Derived Features

Feature/Feature Class	Description
	User Features
$follower\_bucket_i$	Indicator on $followers\_count$ in bucket $i \in [1,11]$. Bucketing polynomial degree: 4, Step: 400
$friends\_bucket_j$	Indicator on $friends\_count$ in bucket $j \in [1,16]$. Bucketing polynomial degree: 2, Step: 5
$listed\_bucket_k$	Indicator on $listed\_count$ in bucket $j \in [1,7]$. Bucketing polynomial degree: 6, Step: 2
$favourites\_bucket_l$	Indicator on $favourites\_count$ in bucket $l \in [1,16]$. Bucketing polynomial degree: 2, Step: 20
$statuses\_bucket_m$	Indicator on $statuses\_count$ in bucket $l \in [1,11]$. Bucketing polynomial degree: 4, Step: 40
	Language Features
sentiment	Binary indicator on sentiment polarity for very positive[0.5,1), positive[0,0.5), neutral(0), negative(-0.5,0] and very negative(-1,0.5]
objectivity	Binary indicator on sentiment objectivity for objective (0.5-1) and subjective (0,0.5]
parts_of_speech_tags	Binary indicator on the presence of each part of speech tag for the number of times that part of speech occurs, (0),[0-5),[5-10),(10+)
	Time Features
$time\_bucket_n$	Indicator on $created\_at$ in bucket $n \in [1,24]$.
	Interaction Variables
$time\_bucket_n \_and\_feature_x$	Indicator on the union of time bucket n and feature $x \in DerivedFeatures \setminus TimeFeatures$

are converted uniformly. We believe this assumption is justified by the traffic pattern we see which matches to PST users. Time is important to our model as after building a regression we predict the optimal input for time.

Note that we do not capture a user's features as node in a graph beyond their degree. We will consider the effects further in the discussion section, but would like to acknowledge that a model that better considers the graphical nature of the Twitter community would better model these factors.

Derived Features

We derive features to overcome three problems with our raw input data: implied linear relationships between real input values and retweets, lack of interaction between input features, and unstructured information in the text. To remove the implied simple linear relation between real valued input variables and output values (predicted retweets) we discretize the range of input variables. We bucket inputs on a polynomial bucketing scheme such that i^{th} bucket for feature class C of size

$$(i+1)^{kC}(bucket\_size_C) - i^{kC}(bucket\_size_C)$$

. We found through experimentation that polynomial feature bucketing out performs linear feature bucketing. We tuned bucket size and polynomial degree for each feature class by plotting the average retweet rate per bucket and tuning the variables to produce a plot with strong signal. Figures 2 and 3 are an example of this process.

Feature interaction between time features and other features is important to our predictive analysis of optimal time because without feature interaction we would predict the same time for all tweets. We took two step to overcome this issue. For linear models we add derived interaction variables by taking the inner product of the time feature vector and the each of the other feature class vectors transpose i.e. define feature set F containing features class vectors V_C, we take interaction variable features

$$\{V_{time} V_{C \in F \setminus V_{time}}^T\}$$

As both vectors in the inner product contain only one non-zero entry (which has value 1) the resulting matrix has a single non-zero entry with value 1.

The language features include: *sentiment*, *subjectivity*, and *parts_of_speech_tags*. We used the Python TextBlob package[3] to help with our Natural Language Processing; it takes a line of text and returns different NLP tools. The first feature we considered is sentiment, which is returned as a float from -1 to 1. We separate it into five binary variables corresponding to very negative sentiment (-1,0.5], negative sentiment (0.5,0], neutral (0), positive [0,0.5), and very positive [0.5,1). Second, we consider subjectivity , which is returned as a float from 0 to 1. We break it into two binary variables for subjective (0,0.5] and objective (0.5,1). Finally, we consider part of speech tagging. TextBlob returns a list pairing

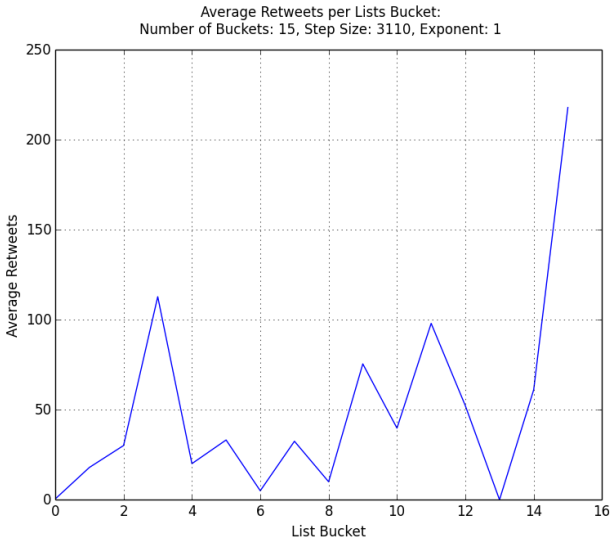

Figure 2: Average Retweet Rate Per User Listed Buckets-Linear Bucketing

each word in the text with its corresponding part of speech tag. We then create binary features for each possible part of speech tag (based on the Penn Treebank Project) and whether there are (0),[0,5],(5-10], or (10+) occurrences of a given tag. We combine these three sets of features to build our complete language model.

Models

Our problem has two main parts to it. First, we need to predict the success (measured as the number of retweets) for a given tweet (i.e. given user info, tweet text, and time). Second, we need to determine the optimal time to tweet given user info and tweet text, i.e. argmax the prediction of retweets over time. We apply four models each an attempt to minimize MSE between predicted retweets and true retweets. We utilized the Scikit-learn Python package[4] to perform the regressions, while we wrote code to process the input and output of the regressions. We also built more complex classifier-regression hybrid moodels around the Scikit-learn classification and regression implementations. Lastly, we'd like to note that Scikit-learn acts as a wrapper to the libsvm library for Support Vector Machines.

Linear Least Squared Regression

We use a linear regression algorithm to predict the success of a given tweet, and then take the argmax over that linear regression to determine the optimal time. Our linear regression model finds the weight vector

$$w = \min_w ||Xw - y||_2^2$$

[3]Loria

[4]Pedregosa

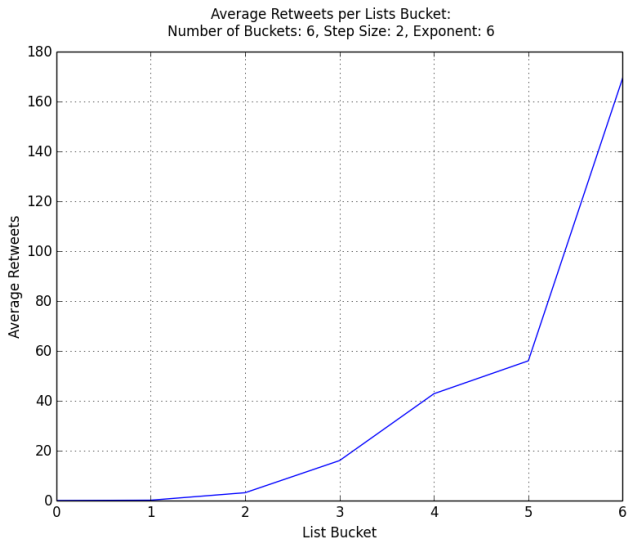

Figure 3: Average Retweet Rate Per User Listed Buckets-Polynomial Bucketing

We found that linear regression was unable to overcome the challenge of highly sparse data well.

Logistic Regression - Least Squares Regression Hybrid

In an attempt to overcome the problems presented by our sparse dataset better than linear regression we applied a regression-classification hybrid. We first attempt to classify tweets as receiving zero or non-zero retweets and then apply a regression model trained on all non-zero retweet tweets to the tweets we classify as receiving non-zero retweets. We use logistic regression to classify zero or non-zero, i.e. we find weight vector w such that

$$\min_{w,c} \frac{1}{2} w^T w + c \sum_{i=1}^{n} log(exp(-y_i(x_i^T w + x))) + 1$$

We then apply the least squares regression defined previously to find a regression output of predicted retweets.

Support Vector Regression

We apply support vector regression to find a more reliable regression model. Our support vector calculations depend on taking:

$$\min \frac{1}{2} ||w||_2^2$$

such that

$$conditions = \begin{cases} y_i - \langle w, x_i \rangle \leq -b\varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases}$$

Support Vector Classification - Support Vector Regression Hybrid

Similarly to the Logistic Regression - Least Squares Regression Hybrid, in this model we again train a classifier to classify tweets as receiving zero retweets or not and then build a regression model on non-zero retweets. In this case we train a support vector classifier on the following dual problem:

$$\min_{w} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \}$$

$$conditions = \begin{cases} y^T \alpha = 0 \\ 0 \leq \alpha_i \leq C, i = 1, ..., \ell \end{cases}$$

Results

We apply the above models to predict the number of retweets for each tweet in our testing set or alternatively in the testing fold of our k-fold cross validation to evaluate our success. We use mean square error as a measure of correctness of our regression, i.e. our error is computed as the average of the square of the difference between the predicted number of retweets and the actual number of retweets. Performing a number of iterations of training on a fixed training set and testing on a fixed testing set we found that our testing error regularly beat out training error. After further inspection we found that large outliers in the training data pulled the training MSE up. Our testing error was not a good test of generalization error. To correct for this we joined our testing an training data and ran k-fold cross validation to estimate generalization error. We found that our regression was able to beat the naive baseline of predicting zero. The naive baseline here is not a trivial bar to beat as about 90% of our data has zero retweets and of those with non-zero retweets very few have values over 10.

Table 3: Training Error

	MSE	Zero MSE	Non Zero MSE
SVR	31.5291	0.0525	166.1153
LSR	30.2451	1.6016	152.7198
SVC-SVR	124.2521	0	124.2521939
LR-LSR	36.7347	0	36.7347
Naive	35.6578	0	184.4605

Table 4: Testing Error

	MSE	Zero MSE	Non Zero MSE
SVR	32.24	0.0889	166.1992514
LSR	33.9613	1.7808	167.0801
SVC-SVR	33.2133	0.3013	170.3319
LR-LSR	1.76E+25	1.2217	8.71E+25
Naive	35.6578	0	184.4605

We visualize our argmax solution to finding the optimal time in Figure 4 by plotting our prediction of retweets for

two real tweets side by side over the course of a day. The points in this plot indicate the true values.

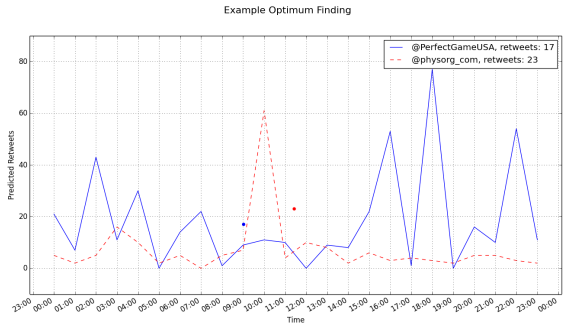

Figure 4: Predicted retweets over time

Discussion

Our data is highly skewed toward tweets that receive zero retweets. This skew has complicated our regression goal. To overcome this obstacle we have worked with two classification-regression hybrid algorithms in which we first attempt to classify 0 vs non-zero tweets and then perform a regression on the non-zero values. This method did not improve our results because the classifier was still struggling to classify correctly because of the sparsity of retweets. We see four primary weaknesses in our approach which may be improved to achieve better results.

Language

The wording of a tweet determines much of its success. Although we employed sentiment and subjectivity classification as well as parts of speech tagging to model the language of a tweet, a human takes much more information to decide whether to click retweet. We struggled to derive wit, humor and apropos qualities of tweets which greatly influence success. With greater computing power it may be valuable to use a bag of n-gram type model of the text to predict on text important to retweets. Another potential solution to extracting more structure from text which would require less computational power is to apply a k-clustering algorithm to the text of tweets and include the assigned cluster as a feature in the regression modeling. This type of solution would hopefully be able identify certain types of tweets which receive more retweets than others.

Graph Structure

Twitter is fundamentally a social network. From user statistics we were able to gauge the outgoing degree of a node (user) in the graph however that is the limit of our treatment of the graph nature of twitter. We believe that further analysis of each user as a node in a graph will yield better results than our method thus far. Questions such as how to represent a user with highly connected followers vs a user with poorly connected followers or how retweets propogate through the network ought to be answered a research continues on this problem.

Evaluation

While MSE is a reasonable evaluator of our regression it does not evaluate our goal to predict optimal posting time. It is difficult to find a good evaluator of optimal time as a user posting a tweet fundamentally changes the audience and could not expect to receive the same reaction twice. We believe the MSE on the regression is a reasonable proxy for evaluating optimal time as a perfect regression for retweets would receive 0 MSE and also perfectly predict the optimal time. It is also important to note that a an imperfect regression can still predict optimal time perfectly as long as the argmax of predicted retweet values is the true optimal time. MSE evaluates to a higher standard than a theoretical evaluator against true optimal time but remains a proxy for an evaluator of our goal.

Regression Goal

Retweets are not a very good estimator of success for two reasons. Tweets are rarely retweeted, about 90% of our dataset has zero retweets. A retweet changes the audience of a tweet. Twitter has recently come out with a new metric called views, which is a count of the number of other users that have seen (ostensibly meaning read) the tweet, which will hopefully be available to the API soon. Utilizing this metric would allow us to further tease apart distinctions between high exposure and high success, i.e. how many people saw it and out of those, how many people felt strongly enough to retweet or favorite it. From this, we would hopefully be able to further separate exposure and virality from popularity as measures of success.

Acknowledgments

We'd like to thank Andrew Ng and all of the CS 229 TA's for their help on this project throughout the quarter. We'd also like to thank Sharad Goel for his MS&E 331 assignment that helped us utilize the Twitter API to collect our data. Finally, we'd like to thank Twitter for their public API that made this project possible.

References

Goel, S. (2014). Assignment 2. *MSE 331: Computational Social Science*.

H. Lakkaraju, J. J. M., & Leskovec, J. (2013). What's in a name? understanding the interplay between titles, content, and communities in social media. *ICWSM*.

Loria, S. (2014). Textblob: Simplified text processing.

M. Tsagkias, W. W., & Rijke, M. de. (2009). Predicting the volume of comments on online news stories. *ICWSM*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Roesslein, J. (2009). Tweepy.