MATH 4432 Statistical Machine Learning Project 3
Self-Proposed Topic: Predicting Trends of Interests in Twitter with Data Visualization Web Application

By LUK, Wing San (20193803) on 20 May. 2018

1. Overview

Understanding how complex it can be for the public to extract practical insights from raw data, I aim to perform a complete trend prediction from scratch and document the details to make the data extraction techniques more accessible by individuals and small companies. The primary objective of this paper is to develop a reliable system to perform trend prediction that delivers promising results and to make the insights accessible for the public, which allows them to have a better understanding on how data analysis works and understand the insights from data.

The first step to a reliable prediction is to have a reliable and quality input data. Therefore, we would start from collecting data, followed by feasibility studies that evaluate and improve the quality of data as input to our research. Then, we would apply different machine learning algorithms, including Linear Regression, Random Forest Ridge Regression, Kernel Ridge regression, K Neural Network, Bayesian Network, to perform trend prediction.

In our research, Twitter will be used as the means of comparison. The result from different approaches will be evaluated against each other to analyse the performance of different models during prediction, with an aim to select the algorithm of best performance in the area of trend prediction. To achieve this purpose, an extensive collection of data must be gathered, and the results must be able to be compared to the work of other researchers. Twitter is chosen as the application domain for three reasons. First, Twitter provides an official API that enables easy and efficient data extraction. Second, the format of a tweet is highly standardized (fixed-character limit, hashtag, etc.) that allows more accurate analysis. Lastly, Twitter is the most popular platform used by researchers in the field of trend prediction. There is an abundant reference and experimental results to compare with.

The outcome of our project consists of two parts: a user-friendly interface and the system backend, which includes algorithms, models and database. In the web-based user interface, users can specify a topic and select one of the implemented algorithms. The system would display the current and predicted trend, with advanced data visualization techniques that allow people to visualize and understand the result intuitively.

Regarding the system backend, several machine learning algorithms are used for the prediction work. First, all the data collected will be processed (normalization, emphasis and smoothing) to convey useful information for analysis. Then, the selected algorithms will be

performed accordingly, showing the predicted trend and the root-mean-square error (or accuracy for classification prediction).

This work is highly results driven. The performance of developed algorithms is our primary focus and will be used to evaluate the effectiveness of the proposed models. An Evaluation Framework, which allows us to display the performance of the proposed algorithms in a lookback window, is developed to provide a fixed setting to evaluate algorithms' effectiveness. The Evaluation Framework, which contains several objective measurements, such as accuracy, coverage and precision of the prediction, is a powerful tool that allows continuous evaluation and improvement of the system backend.

2. Related Work

2.1 Data Preprocessing

Having the goal of developing a reliable system to perform trend prediction that delivers promising results, it is of crucial importance to assure the quality of input data. In addition to crawling data from a reliable source, there are techniques that can be adopted to boost the performance of prediction algorithms by applying different transformations to the time series to pre-process the input data[2].

In [2], a three-standard transformations, where each captures a different type of underlying similarity in the data, has been proposed by Anthony, Luke, Jon and Jason. The result shows that the classification performance can be improved by applying different transformations to the time series beforehand. It is also supported by another paper that applies a similar technique[3].

2.2 Learning Model

There have been numbers of machine learning approaches to performing trend prediction, where many of them achieved a significant result. This paper would reference some of the prediction models used in other papers and compare the result against each other to determine which of them perform the job of trend prediction the better.

Nearest-Neighbour Approach

In [3], Shah and George cluster similar time series into groups, with each group representing a unique pattern of popularity against time. To perform prediction, a new set keywords/ hashtags will be compared with the clustered data to match the best-fit pre-defined data patterns.

Binary Classification Approach

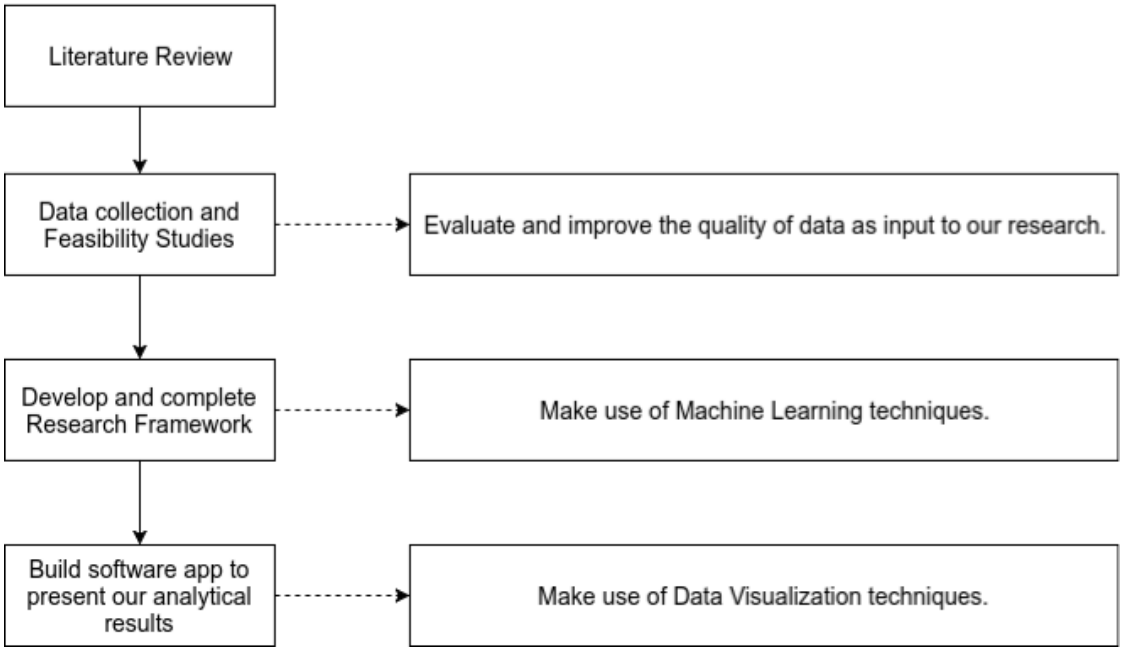
In [4], the prediction task is modeled as a machine learning classification problem. The classifier is trained using data collected over one day and is able to distinguish between trending hashtags and non-trending ones. The result of the binary trending/ non-trending classification achieves very high precision and reasonably high recall in identifying trending hashtags.

Linearity and State-Space Model

In [5], Peng, Xufei and Baoxin approach the prediction task by designing a combination of (non-)linear and (non-)state-space prediction models. The result provides information into the advantages and disadvantages of different properties of a model.

The presented researches demonstrate some existing approaches to apply machine learning to perform twitter trend prediction. Different researches provide insights into the specific algorithm handling the prediction task. However, there is lack of a means to compare their strength and weakness directly. Furthermore, research papers tend to focus on one particular issue, instead of the complete process of prediction. This paper aims to learn the algorithms used and provide a means to compare them with other algorithms. Ultimately, through combining with data visualization techniques, we would develop a reliable system to perform trend prediction that delivers promising results which can be understood by people easily.

3. Project Design



The scope of our project mainly consists of 3 phases. We started by collecting, evaluating and improving the raw data available to us. Then, we developed and implemented a research

framework to structure our data in a way that is best for data analysis. Lastly, we developed a scalable software, together with suitable data visualization techniques, to make the results accessible to the interested audience.

4. Data Collection and Feasibility Studies

4.1 Data Extraction

Streaming tweets and trends from Tweepy

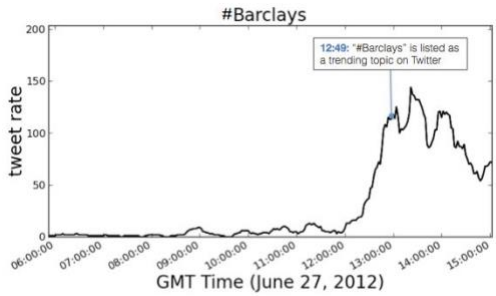
To collect tweets needed for our research, we use Tweepy, a python package interfaced with Twitter API, for fast and easy development. Since the streaming rate of tweets is limited, we need to narrow the source geographically so that the concentration of the tweets, when partitioned in time intervals, is sufficient for prediction algorithms. London is a suitable candidate for our research because it is an English speaking city with the highest share of tweets per day (2% or 10 million of 500 million globally). We design a square box region with (latitude, longitude) pairs equaling [51.27, -0.54, 51.7, 0.3] to capture tweets from Greater London Area so that the contents are more concentrated than if we stream from globally. For streaming trends, we collect the trending list of topics and hashtags with Twitter API's trends_place function with London's Where On Earth IDentifier(WOEID), 44418, as the filtering argument.

In order to reduce redundancy of the data, we collect only the fields of tweets that are suitable for analysis. For trending information four fields are included: *name, query, tweet_volume and created_at.*

4.2 Data Quality Analysis

Data quality of trending information

It could be inferred from [3] that a hashtag could be officially declared trending if the count of relevant tweets rises rapidly above the average of previous counts over a period of time, as illustrated in the following figure (2min time bucket). Such trends can be captured with a 3-hour window centered at the time of trend declaration. Such samples can be analyzed, as done by previous researchers.



To validate the usefulness of the data we extracted, we look at trending hashtags with highest tweet counts streamed and their respective time series of tweet counts before the trend was declared, represented by 3h time buckets to reduce processing time. If the dataset is useful, it should be clear that

1: There is an obvious increase of volumes at the time the hashtag was declared trending, and

2: there are sufficient volumes, albeit relatively lower, before the trending spike, for input data of our analysis.

We have 1000 pieces of trending information with the top hashtag being “halloween” containing 2000 tweet counts before the declaration. Other 950 trending tags are with tweet counts below 100. We select the top 50 hashtags for observation.

The result, as shown in the figure below, reveals that the tweets and trending information streamed are not synchronous with each other. Except for the first hashtag, the rest either have very noisy tweet counts with no noticeable trend or very low tweet counts followed by a sudden spike, which is nowhere near the time of trending declaration. This is because twitter

API offers limited sample rates, which is done in real time, which implies that it is not possible to access the full tweet history of specific trending hashtags synchronized with trending information without commercial upgrade.

Sample of time series before official trend declared (3hour time bucket). After that counts are set to zero

To cope with the deficiency in data quality, we need to:

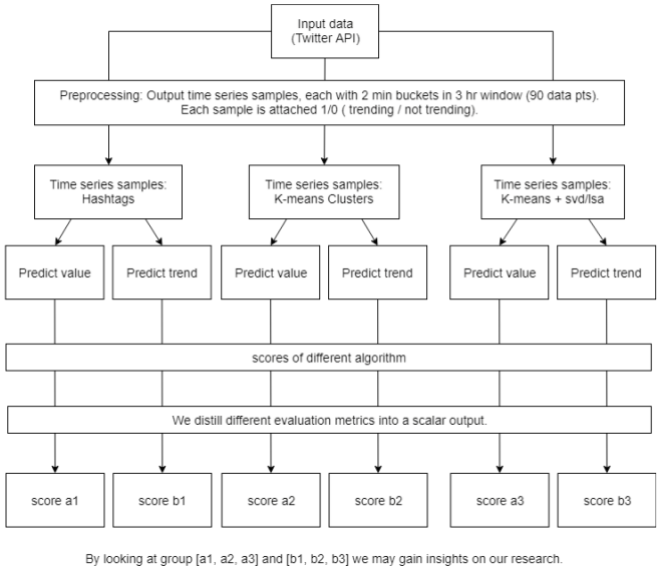
1. Define whether the time series segment is trending/ not trending, instead of relying on information from Twitter API

2. Make sure the time series segment we need contains sufficient data count.

It can be done as outlined by specifying a rolling 3-hour window on all time series, then extract them if the data count inside the window is over 200. Those samples are deemed valid and assigned 1(trending) or 0(not trending) as defined in our research framework.

5. Research Framework

5.1 Overview



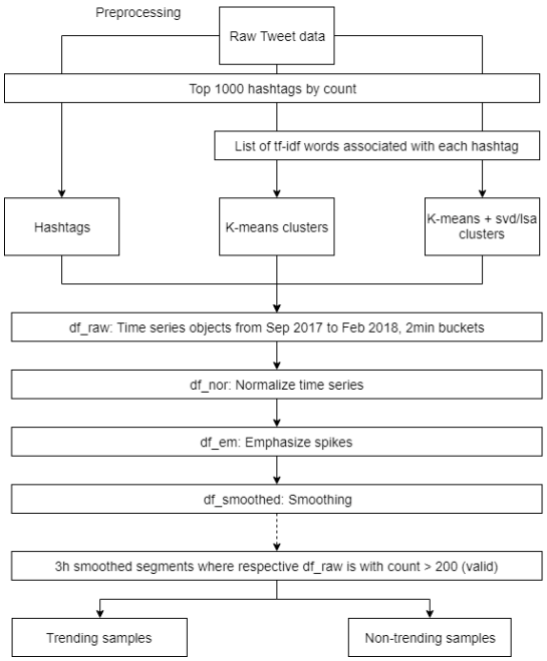
Under our research framework, 3 categories of time series data, namely time series per hashtag, k-means clusters and k-means + SVD/LSA clusters, will be analysed. For each category, we extract samples of 3-hour segments for analysis. The segments are selected in range [t, t+90] and assigned a trending / not trending score based on information

at time = t + 91. This ensures that we are predicting the trend before it actually happens. Our framework is structured so to find out:

3. Whether a certain algorithm is better at predicting value/trend.

4. Whether by clustering tweets can the time series be better predicted.

5.2 Preprocessing



An overview of our dataset:

Number of raw tweets: 10398733

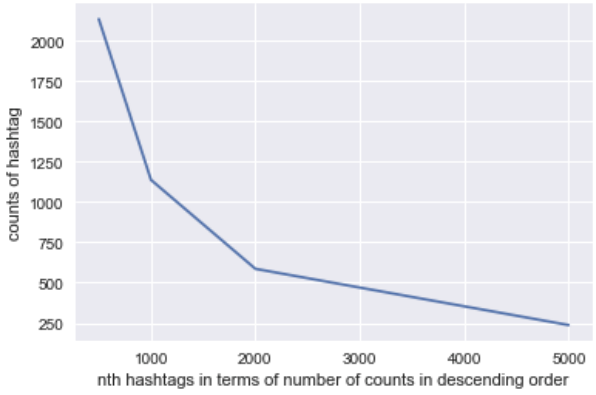
Number of hashtags: 552460

	Count of tweets that contains the hashtag
500th hashtag	2134
1000th hashtag	1138
2000th hashtag	586
5000th hashtag	237

To ensure each time series contains a few suitable segments, we select tweets associated with top 1000 hashtags by count. The reason why the top 1000 hashtags is chosen, instead of top 1500 or top 2000 hashtags, is that it is clear that the number of counts of hashtags dropped exponentially from the graph below. Therefore, hashtags after top 1000 in terms of counts are believed to be very difficult to measure and can be neglected.

Categorizing data with k means To generate time series per hashtag, we collect all timestamps of tweets associated with the hashtag and extrapolate them on to a time series.

To produce time series per cluster, we first calculate term frequency-inverse document frequency(tf-idf) on aggregated texts associated with each hashtag.



TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Below is the graph showing the formula of tf-idf we used[6]. In here, “document j” in the graph below means those tweets associated with the hashtag, and the number of documents simply means the total number of tweets we collected.

Then each hashtag is attached with a list of tf-idf words sorted by tf-idf score. For example, the hashtag #crypto will have a list ['crypto', 'bitcoin', 'blockchain', 'coincheckup', 'daytrader' ... ‘some’, ‘thanks’, ‘my’]. We select the top 5 sorted tf-idf words of each hashtag as input to our clustering algorithms.

After that, we mix the top 5 sorted tf-idf words with the respective hashtag, grouping them into a list. For example, 'ht_name' represents hashtag from the json object {'ht_name': 'crypto', 'sorted_tfidf_words': [['crypto', 0.184], ['bitcoin', 0.066], ['blockchain', 0.05], ['coincheckup', 0.037], ['daytrader', 0.037], ...] , which is “crypto”. Its corresponding top 5 sorted tfidf words are crypto, bitcoin, blockchain, coincheckup and daytrader respectively. As a result, we group these 5 words with the hashtag together to become a list of [‘crypto’,‘crypto’,'bitcoin','blockchain','coincheckup','daytrader']. Due to the fact that some of the top 1000 counts hashtags are basically the same with differences only between capital letter and small letter (e.g. “Hong Kong” and “hong kong”), we grouped these hashtags together, and it comes out to be 562 top counts different hashtags. As a result, we repeated the step for all the top 562 counts hashtags by grouping them with their relative tfidf words. As a result, we collected 562 lists of words.

We have used the open-sourced library Sklearn for doing clustering of words from hashtags and tweets. Here, we first remove all stopwords (e.g. is, a, am, are, etc) from the lists of words. Then, we convert the list of words to a matrix of TF-IDF features again. After that, we set the number of clusters we would like to get from kmeans, which is an unsupervised machine learning method. Here in the code, we tried to make 300 clusters from the original lists of words we have, with each consisting of 2 words. In fact, we have also tried to make 300 clusters with 4 words in each cluster and 300 clusters with 3 words in each cluster. We have also tried and make txt files with 562 clusters in which each cluster contains 4 words, 3 words and 2 words respectively. We have repeated these different combinations of clustering in the clustering with singular-value decomposition (SVD) and l atent semantic analysis (LSA) used before applying K-means for dimension reduction in order to reduce noise (The methodology of this clustering with SVD and LSA would be explained later). However, the number of trend = 1 discovered from those 562 clusters in a different time is quite similar, no matter each cluster contains 2 words, 3 words or 4 words.

	number of sample of cluster with trend =1
cluster_562Cluster2	66
cluster_562Cluster4	62
cluster_withsvd/lsa_562clusters_2	43
cluster_withsvd/lsa_562clusters_4	42

For the table above, cluster_562Cluster2 means 562 clusters with each cluster having 2 words by k means only, cluster_562Cluster4 means 562 clusters with each cluster having 4 words by k means only, cluster_withsvd/lsa_562clusters_2 means 562 clusters with each cluster having 2 words by k means, SVD and LSA and cluster_withsvd/lsa_562clusters_4 means 562 clusters with each cluster having 4 words by k means, SVD and LSA.

Moreover, for a smaller number of clusters (e.g. 300 clusters), a smaller number of trend =1 was found. Since we need a dataset of clusters with an adequate number of trend =1 for training the model, we used the dataset with 562 clusters (each cluster 3 words) to train our machine learning models and used for accuracy comparison afterwards. We did the same thing for SVD and LSA with K means.

A Sample of Clusters by K-means only:

Top terms per cluster: Cluster 0:

カナやん

西野カナ

Cluster 1: bigdata ai

Cluster 2: auspol ozloop

Cluster 3: 마이크드랍 아뭬멜뮵툡툡

Cluster 4: edmfamily Twtr

....

Cluster 299: holidays happyholidays

Categorizing data with LSA and SVD before applying k means

This time, we apply SVD first after vectorizing the list of words and converting the list of words to a matrix of TF-IDF features. Then, we normalize the vectorizer. At the end, we apply LSA to the normalized SVD vectorizer. Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the totality of information about all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and set of words to each other.[8]

All these steps are done in order to reduce dimensionality and make K means to behave as spherical k-means for better results. Then, we apply k means again for text clustering.

Statistics for the clustering with using LSA and SVD before k means (561 clusters, each with 4 words):

Homogeneity: 0.796 Completeness: 1.000 V-measure: 0.887

Adjusted Rand-Index: 0.000 Silhouette Coefficient: 0.387

A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class. At the same time, a clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) when compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. From this example and the statistics above, we know that this method cluster the text appropriately.

A sample of clusters by LSA and SVD + K-means (561 clusters, each with 4 words):

Top terms per cluster:

Cluster 0: filtering 마이크드랍 bts_ant 황민현

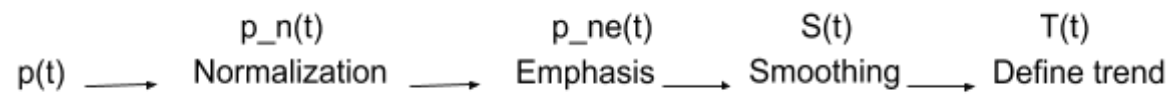
Cluster 1: mysex sistersex freeborn 3some

Cluster 2: daily thanks latest girlstwerking

Cluster 3: garden diy scapamor88 cozy ....

Cluster 561: time giveaway funny funnyvideos

Preprocessing on time series



The raw rate $p(t)$ counts the number of tweets in time bucket t spanning 2 minutes

each. We normalize the count by: $p_n(t) = p(t)/\ mean(\ p(0, 1, \ldots, t)\ )$. Large spikes are

emphasized:

$$p_{ne}(t) = \left| p_n(t) - p_n(t-1) \right|^a \quad \text{, where } a=1.2$$

Then we smooth the *signal, where*

$$\text{signal: } S(t) = \sum_{i=0}^{4} p_{ne}(t-i).$$

Finally, we define a trend as:

$T(t) = S(t) / avg(S(0, 1, ..., t-1)) > 3$, and non-trend otherwise.

Since trending activity is characterized with a spike above the average of historical rate. We normalize the series by dividing it with average of all counts leading to time t . The side effect is that the rates close to the spike will be amplified so that more noticeable patterns may evolve before the trend is declared.

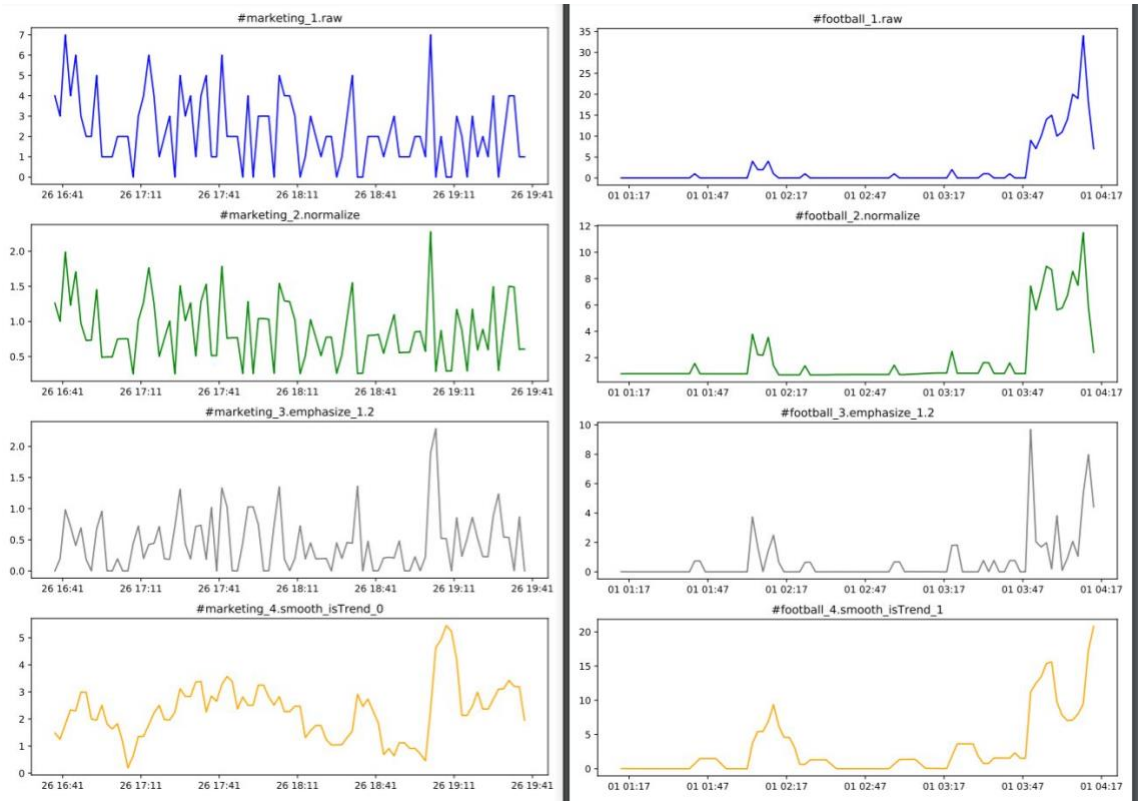
As trending activities are impulsive, instead of stable uptrends, we emphasize the spike by amplifying the change of tweet rate by a parameter $a \geq 1$. we use a = 1.2 as suggested by [3].

Then we smooth the signal with a cumulative sum of rolling 5 time buckets / 10 minutes.

Finally, using results derived from previous researches[3], we define trend if the tweet rate is 3 times over the average of all previous rates within the sampling window(3h long). We reasoned a factor of 2 is not enough to display substantial spikes to the upside and a factor over 3 may limit the detections of many noticeable trends. Some extreme values, such as a multiplier over 10, would actually allow negligible spikes with rates close to zero preceding it, meanwhile blocking the large spikes that are immediately observable.

When all outputs are generated, we place a rolling 3h window on the raw rate p(t + 1). If within the window the count is over 200, this segment is considered valid. We then extract S(t-89 ... t) as an sample and assign attribute isTrend=1 or 0 depending on T(t+1). Since the trend value of trending patterns stays at 1 for several data points, after selecting the sample, we will not look at time = t +1, t+2 ... t+90 as to avoid duplicating sample of the same trend.

Example output: isTrend: left sample=0 ; right sample=1



5.3 Predicting Values of the Samples

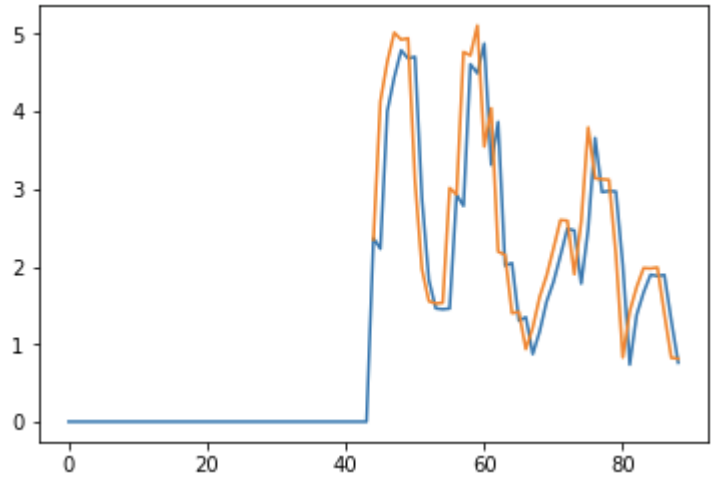
In this section, we seek to predict $S(t + 1)$ using parameters in $S(t, t - 1)$. For each sample, we train the first half of the data to test the second half of the data. Then we will take an average on the root-mean-square error on all samples on each algorithm, in order to evaluate the effectiveness of them.

Experimental results:

We applied 7 machine learning algorithms in predicting smooth values on all samples. After that, we applied the machine learning algorithms to predict the second half of smooth value of a sample by using its first half for training the model.

For example, the graph below shows the predicting result of smooth values of the sample hashtag “crypto” with time started on 2/6/2018 14:56 by kernel ridge regression. The Orange line represents the predicted smooth values, while blue line represents the real smooth value. The y-axis is the smooth value, while the x-axis is the time axis. The root-mean-square error is calculated for each sample prediction, just like the one below.

	hashtag	kmeans	kmean_LSA/SVD
sample size(valid 3h samples)	2415	1698	1267



Below is part of the dataframe object storing each individual hashtag with different accuracy we calculated from different machine learning methods:

Method \ RMSE	hashtag	kmeans	kmean_LSA/SVD	mean
K-neighbors regression	6.08	1.30	1.16	2.85
Random forest regressor	6.13	1.26	1.14	2.84
***Gaussian process regression**	**344.36**	**321.28**	**282.80**	**316.15**
Decision tree regressor	6.32	1.41	1.27	3.00
****Linear support vector regression**	**4.46**	**0.74**	**0.66**	**1.95**
Multi-layer perceptron neural network	6.19	1.19	1.12	2.83
****Kernel ridge regression**	**4.45**	**0.76**	**0.67**	**1.96**
median	6.13	1.26	1.14	
mean	54.00	46.85	41.26	

In [117]: df

Topic	Time	t-1	t-2	predict	kernel_ridge_regression_accuracy	DecisionTree_regression_accuracy	KNN_regressio
#crypto	20180206-145600	[5.866781650063625, 6.942181615179211, 7.91680...	[6.942181615179211, 7.916803738260816, 7.91680...	[7.916803738260816, 7.916803738260816, 7.91680...	0.781471	1.406661	
#momlife	20171004-125800	[1.6887096020309584, 1.669306283127165, 1.6693...	[1.669306283127165, 1.669306283127165, 1.66930...	[1.669306283127165, 1.669306283127165, 0.83837...	3.547263	14.255697	
#momlife	20171004-160000	[32.99715420051254, 20.656776015416707, 14.176...	[20.656776015416707, 14.176927376945308, 5.640...	[14.176927376945308, 5.640454927649346, 7.8619...	0.306519	0.354609	
#momlife	20171005-165600	[4.461275323395318, 4.431003144423344, 4.43100...	[4.431003144423344, 4.431003144423344, 3.75689...	[4.431003144423344, 3.7568933267419085, 3.0827...	2.894655	9.823598	
#momlife	20171005-195800	[25.81707915330637, 16.21734151050693, 6.72544...	[16.21734151050693, 6.725441562463121, 3.91973...	[6.725441562463121, 3.9197397199867248, 4.2780...	0.213084	0.599260	
		[2.9195127184699343,	[2.1897362803496367,	[1.452271262215291,			

A summary of the root-mean-square error of the prediction on each sample in mean and median are presented below:

***Worst predictor **Best predictors**

There are three important findings that are critical to this section. Firstly, the Gaussian process gives the worst prediction rate across all categories of data, with

average rmse around 300. Secondly, Linear support vector and Kernel ridge are the best predictors, with average rmse around 1.95. Finally, we can see that by performing k-means clustering on our tweets, the median rmse is reduced from 6.13 to 1.26, which can be further lowered through SVD/LSA optimization.

Discussion

Overall, Gaussian Process is not a stable predictor. By looking at the samples which GP gives the worst rmse (over 10000), we can see that those samples often consist of sudden spikes:

	#bestfanarmy	#goldenglobes	#btsarmy	#thankyoubts
	20180110-061400	20180108-104000	20180208-190600	20171210-150000
Kernel ridge regression	0.45	8.34	0.58	0.53
DecisionTree regression	2.64	23.13	0.90	1.26
KNN regression	1.51	15.08	1.18	1.02
MLP neural network regression	1.90	6.83	0.80	0.91
Random Forest regression	1.81	15.25	0.64	0.90
Gaussian Process regression	16,027.71	12,019.80	10,460.31	9,441.70
Linear Support Vector regression	0.46	10.55	0.58	0.49

Before we discuss why Gaussian Process Regression performed poorly, we should understand how Gaussian Process Regression works first. Gaussian Process is implemented for regression purpose here. It assumes a gaussian uncertainty on the y-values for prediction.

It is believed that Gaussian Process performed badly mainly because of its gaussian assumption. As we are doing trend prediction, most of the time the y-value (smooth value in this case) will be low and may suddenly rise sharply when the word becomes trend or after some big news. This is definitely not a gaussian distribution and so may affect the prediction. Moreover, Gaussian Process Regression has a low flexibility as it is confined by normal distribution. As a result, it may easily be affected by outliers once they do not follow the normal distribution.

On the other hand, Linear Support Vector and Kernel Ridge are very good predictors, even when dealing with volatile dataset. They are kernel based regression models which do not offer a probabilistic interpretation, unlike Gaussian Process Regression. Linear support vector machine is also effective in high dimensional spaces. It is still effective in cases where

number of dimensions is greater than the number of samples, unlike gaussian process which performs very bad in high dimensional spaces. Kernel Ridge regression is similar to linear support vector with just using different loss functions. KRR uses squared error loss while support vector regression uses epsilon-insensitive loss. Given the high dimensional spaces required from our complicated data, they provide high accuracy with low RMSE in average.

Finally, we can see that by clustering the tweets, the prediction accuracy can be improved. We want to examine the effect of them by calculating the variance on rmse of each algorithm on the samples:

Method \ Variance of rmse	hashtag	kmeans	kmean_SVD/LSA	(kmeans_SVD/LSA) / kmeans) - 1
K-neighbors regression	105.44	8.17	7.65	-6.36%
Random forest regressor	105.67	8.15	7.75	-4.94%
Gaussian process regression	674,218.62	355,812.84	381,908.38	**7.33%**
Decision tree regressor	105.86	8.35	8.10	-3.01%
Linear support vector regression	79.40	2.89	3.25	**12.61%**
Multi-layer perception neural network regression	183.63	8.79	11.24	**27.87%**
Kernel ridge regression	79.49	2.90	3.17	**9.23%**

It is clear that by clustering the tweets, the variance of rmse can be drastically reduced, leading to better prediction result. Since the variance of rmse is far smaller by k means clusters than individual hashtags in every machine learning algorithm, it is believed that there are far fewer outliers of sample rmse for the clustered sample by k means and k means with SVD and LSA.

Applying k means for word clustering is useful as expected which led to better prediction results because after applying k means outliers were greatly reduced. The number of samples was greatly reduced after applying k means clustering, and similar words tended to be within the same cluster. As a result, words which were hot topics tended to have trended words in their clusters too as people usually wrote similar words for particular trend topics. Results, therefore, became very condensed, which words with a high smooth value having a cluster with even higher smooth values after summing up several high smooth value words together. On the contrary, a word which was not in trend seldom had people write their similar words in tweets. As a result, their cluster tended to have low smooth values given the words in the same cluster also having low smooth values. Differences of smooth values between trending cluster and the non-trending cluster would be huge, and this helps improve accuracy for training the machine learning models, as well as making a prediction. Root-mean-square error therefore greatly reduced after clustering.

Also, it is worth noting that SVD/LSA optimization slightly increase the variance of rmse, while the average of rmse is still better than that of k-mean only clusters. It is believed that this is due to the noise reduction by SVD and LSA. What SVD and LSA do are mainly for reduction of dimensionality, helping the k means algorithm to group similar words together easier. As a result, more similar words would be grouped together comparing to only using k means, and so for the same reason mentioned above the prediction should be more accurate. RMSE, therefore, became lower. The reason why the variance for this method was slightly higher than only using k means might due to the fact that some samples were reduced too much in dimensionality and could not be distinguished with other samples. As a result, this might also slightly affect the efficiency of K means method and produced some outliers, leading to outliers in rmse after making predictions.

5.4 Predicting trend/non-trend of the Samples

In this section, we seek to predict T(x) using S(y) for $y < x$. Since certain patterns are evident before the official trend detected by twitter, if we can recognize the patterns preceding our custom defined trend, our method could be proved to work on official data set.

In our programming. We select a point time=t+1, which data count is enough, as T(x), we then extract the sample window time=(t-89 to t) as S(y), and assign binary trend/non-trend to the sample.

For each algorithm, it will output the confusion matrix and ROC curve showing the

prediction accuracies. Since the data contains a lot of samples where isTrend=0, all algorithms obtained very high false negative rates on them (over 99%). Therefore, we are mostly concerned with the true positive rate.

	hashtag	kmean	SVD/LSA
num of input hts/ clusters	2000	562	564
# valid 3h samples (1)	2415	1698	1267
# samples isTrend=0	2008	1632	1228
# samples isTrend=1 (2)	407	66	39
(2) / (1)	16.9%	3.9%	3.1%
# testing samples isTrend=1 after k-fold	41	6	4

We have applied cross validation methods to ensure the prediction and the measured accuracy would not be bias based on particular testing sets and training sets. For example, we applied k-fold cross validation, with k = 10 before running our machine learning algorithms.

There would be k experiments conducted each time, which means we would run the respective machine learning algorithm k times. Each time, the testing set and training set would be different. In our case, we divided our dataset into 10 pieces, each time picked one piece of the dataset for testing and the other 9 pieces for training. This process would then be repeated 10 times to ensure the whole dataset is covered for both training and testing. In each experiment, we also applied stratified sampling to ensure there would be same proportion between the sample with trend = 1 and trend =0 in both training set and testing set. For example, if there are 2 samples with trend = 1 and 18 samples with trend = 0, then we want to split them into half, one for training set and one for testing set. Then, we would get 9 samples with trend = 0 and 1 sample with trend=1 in both training set and testing set. This is to ensure the machine learning model can be trained probably with good feeding data. Otherwise, there would be chances that the whole training set contains sample with trend = 0 and whole testing set with trend = 0 and trend =1, and results would be seriously affected. It is especially important to apply stratified sampling in our case given extremely imbalance proportion between Trend =1 sample and Trend = 0 sample shown in the table above. After the splitting and distribution of samples into training set and testing set, we run our machine learning

algorithm and did a confusion matrix for each experiment, i.e we did 10 times of confusion matrix, each for each experiment inside the 10-fold validation. We also did a ROC curve 10 times. We calculated out the average true positive rate among all experiments inside the 10-fold validation, as well as the maximum true positive rate among those 10 experiments.

Experimental results:

Method \ True Positive rate	hashtag		kmeans		kmean_ SVD/LS A	
	mean	max	mean	max	mean	max
Multi-layer Perceptron Neural Network Model - 11 hidden layers	0.94	1.00	0.56	1.00	0.63	1.00
Gaussian Naive Bayes	0.93	1.00	0.59	0.86	0.51	1.00
Gradient Boosting Classifier	0.91	1.00	0.61	0.83	0.47	0.75
Decision Tree Classifier	0.91	0.95	0.70	0.86	0.41	0.75
Regularized linear models with stochastic gradient descent (SGD)	0.89	1.00	0.46	0.86	0.49	1.00
Two-class AdaBoost boosting algorithm using AdaBoost-SAMME with decision trees	0.89	0.95	0.65	1.00	0.53	1.00
Logistic regression	0.89	1.00	0.50	0.71	0.41	0.75
Random Forest Classifier - max depth = 15	0.88	1.00	0.50	0.86	0.31	0.50
Linear Support Vector Classifier	0.86	0.95	0.53	0.86	0.39	0.75
Passive Aggressive Classifier	0.84	0.98	0.53	0.86	0.36	0.75
K-neighbors classifier	0.82	0.90	0.42	0.71	0.42	0.67
Extra-trees classifier	0.79	0.93	0.36	0.67	0.38	0.75
Bagging + k-neighbors classifier	0.77	0.90	0.32	0.67	0.28	0.50
Gaussian process Classifier	0.75	0.85	0.36	0.57	0.13	0.50
Nearest centroid classifier	0.63	0.73	0.59	0.86	0.54	1.00
using Ridge Classifier	0.52	0.66	0.36	0.83	0.20	0.50
Bernoulli Naive Bayes	0.44	0.53	0.16	0.43	0.27	0.67
Label Propagation classifier	0.30	0.37	0.05	0.17	0.05	0.50

Results show that Multi-layer Perceptron Neural Network beats other algorithms in terms of recognizing the patterns of trending behaviors (94% TP rate), followed closely by Gaussian Naive Bayes, Gradient Boosting and Decision Tree (91-93%). Label Propagation is the worst predictor of all, with average TP rate 30%.

Also, after clustering the tweets, the prediction rate drops drastically. It could be due to the lack of sample size. However, the low ratios of number of sample(isTrend=1) vs all valid samples provide us with some insight to this problem. Since the rate is low as 3-4%, increase the size of raw tweets will not make this method of analysis more efficient.

Discussion :

From the result, we can see that MLP neural network and Gaussian Naive Bayes perform very well.

MLP here was set with 11 hidden layers, each layer with 11 neurons. It performed well due to its high flexibility for learning and capability to learn non-linear models. Since our data was complicated, this gave an edge to neural network model like MLP to provide high accuracy prediction.

Gaussian Naive Bayes implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters $\sigma_y$and $\mu_y$are estimated using maximum likelihood.

This model performed well since we have applied stratified sampling to ensure all training and testing sets have balanced trend=1 and trend=0 samples. As a result, Gaussian likelihood of features was maintained in the testing set.

Among all models, Label Propagation performed the worst. Label Propagation is a semi-supervised machine learning algorithm that assigns labels to previously unlabeled data points. At the start of the algorithm, a (generally small) subset of the data points have labels (or classifications). These labels are propagated to the unlabeled points throughout the course of the algorithm[9]. Since there are many trend= 0 samples, it is very likely that the data points selected from the beginning are all trend= 0 samples by this algorithm. As a result, very inaccurate prediction may be resulted when the labels propagated.

The low rate of output samples with isTrend=1 can be explained by the smoothing effect of clustering. Since cluster aggregating the counts of its cluster members to a time series. The volatility of the time series will be lower than that of individual hashtag, so that the number of samples that is trending (with its value over 3x rolling average) is reduced.

Yet, it is possible to examine the effect of clustering on the hashtag samples that are trending. We tried to count the number of samples having isTrend=1 inside every cluster. For example,

hashtag	count
a	2
b	1

c	0
cluster ['a','b','c']	3

Our results:

	kmeans	kmean_SVD/LSA
total num of cluster	562	562
num of cluster having its members having isTrend=1 samples	89	96
num of samples of its members having isTrend=1	188	410

It is clear that kmean+SVD/LSA clustering is able to capture more samples of individual hashtags that are trending. It suggests that SVD/LSA optimization is more capable of recognizing important keywords that will attract more mentions, thus contribute to trending behavior.

5.6 More on Clustering

In fact, we have also tried another clustering method which is to conduct t-SNE before applying k means. t-SNE is another dimensionality reduction method similar to SVD and LSA.

We did the clustering, and also the preprocessing of the clusters to generate sets of respective smooth value. After that, we picked the two machine learning methods with the lowest RMSE to do regression on those clusters for prediction. Similar rmse were obtained comparing to using SVD and LSA. Therefore, we did not do further investigate on the clusters since we expected that similar results would be obtained from other machine learning

algorithms.

rmse	t-SNE+kmeans
Kernel Ridge Regression	0.66
Linear Support Vector Regression	0.6498

6. Conclusion

In conclusion, we have identified the pitfalls of directly using the free Twitter API streaming service to conduct data science analysis, as evident in similar previous projects. We have made improvements on the raw data by extracting tweets with sufficient volumes and processing them into structures suitable for analysis. We also defined the trending condition of topics to remedy the lack of complete trending information provided by Twitter API.

By evaluating the data with different algorithms, those performing well in our dataset are algorithms that deal with higher dimensions and uncertainties, while those performing bad are often algorithms that are not flexible enough to recognize new changes in dataset. Also, we have found that by clustering tweets into groups of topics, the prediction accuracy on the rate of interest can be greatly enhanced, and by further optimizing the clusters, we can capture more topics that would meet the trending condition.

Combined with our analytically results, we have built software app to perform data visualizations so that users and derive insights from mass scale user generated contents in an intuitive and interactive manner.

7. Reference

[1] K. Karp, “New research: The value of influencers on Twitter,” Twitter, 10-May-2016. [Online]. Available: https://blog.twitter.com/marketing/en_us/a/2016/new-research-the-value-of-influencers-on-tw itter.html. [Accessed: 10-Apr-2018].

[2] A. Bagnall, L. Davis, J. Hills, and J. Lines, “Transformation Based Ensembles for Time Series Classification,” Proceedings of the 2012 SIAM International Conference on Data Mining, pp. 307–318, 2012. [3] G. Chan, S. Nikolov, and D. Shah, “A Latent Source Model for Nonparametric Time Series Classification,” thesis, MIT Press., 2013.

[4] Das A., Roy M., Dutta S., Ghosh S., Das A.K. (2015) Predicting Trends in the Twitter Social Network: A Machine Learning Approach. In: Panigrahi B., Suganthan P., Das S. (eds) Swarm, Evolutionary, and Memetic Computing. SEMCCO 2014. Lecture Notes in Computer Science, vol 8947. Springer, Cham [5] P. Zhang, X. Wang and B. Li, "On predicting Twitter trend: Factors and models," 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), Niagara Falls, ON, 2013, pp. 1427-1429. [6] C. V. Nicholson and A. Gibson, “Bag of Words & TF-IDF,” Bag of Words - TF-IDF - Deeplearning4j: Open-source, Distributed Deep Learning for the JVM. [Online]. Available: https://deeplearning4j.org/bagofwords-tf-idf. [Accessed: 10-Apr-2018]. [7] C. Manning, P. Raghavan, and H. Schütze, “Cluster cardinality in K-means,” K-means. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/k-means-1.html. [Accessed: 10-Apr-2018] [8] T. Landauer and S. Dumais, “What is LSA?,” What is LSA? [Online]. Available: http://lsa.colorado.edu/whatis.html. [Accessed: 10-Apr-2018]. [9] X. Zhu and Z. Ghahramani, “Learning from Labeled and Unlabeled Data with Label Propagation.” [Online]. Available: http://mlg.eng.cam.ac.uk/zoubin/papers/CMU-CALD-02-107.pdf. [Accessed: 10-Apr-2018].