UNSW

Australia

School of Computer Science and Engineering

The University of New South Wales

Twitter Retweet Prediction

Thesis submitted as a requirement for COMP4121 final project

Supervisor: Prof. Aleks Ignjatovic

Abstract

In recent years, the social network plays an important role in information diffusion. Under such circumstances, for social network companies, such as Twitter, Facebook, it is critical to store the data efficiently and advertising products based on information diffusion rate of different categories. As a result, we designed a reliable, feasible and real-time updating solution to predict the forwarding volume of social network messages. The solution based on iterated filtering and recurrent network (topic classification) has been proposed in order to predict the number of retweets on collected Twitter retweeting data.

Contents

Introduction

As social network plays an important role in the internet era, analysis and prediction of user behaviors will optimize commercial decisions. There are over 330 million active users on Twitter and even US president announces the first-hand news on this platform. Successfully predicting retweets can enhance data storage and refine marketing strategies. There are various algorithm for retweets prediction (Hong, et al., 2011; Zaman, et al, 2014). However, accurately predicting retweets is still challenging. One of the major challenges is the existence of many “robot” followers (directly followers but rarely retweet) on Twitter. Simply counting user’s followers will not provide the most accurate result. It is more desirable to rank users based on their information diffusion rates, which are the amount of retweets. Moreover, different users have different “flavors”, hence proper content analysis is necessary for studying retweeting behavior.

In this thesis, the literature review section will briefly summarize recurrent network (long short term memory), voting algorithm and some previous retweeting prediction algorithms. The subsequent section will present our solutions. To test the accuracy of our algorithm, implementations and evaluations are necessary. The result is evaluated by calculating errors on different sample sizes and different time intervals.

Literature Review

Retweeting Analysis based on Epidemic Model

This approach uses the epidemic disease model as the prototype to predict the retweeting of tweeters. The model is trained by supervised learning. A simple epidemic model divides a crowd of people into three groups: infectious people susceptible people and recovered people. In a tight group of people, every person has the chance to be in contact with another person. A susceptible person may be infected by an infectious person. $S(t)$ is the number of direct followers of all retweeters at time t. $I(t)$ represents the number of retweeters at time t. $E(t)$ represents the number of external visitors who have spontaneously retweeted the tweet at time t. All retweeters have a transmit rate of β of getting their followers retweet their messages. After retweeting, all retweeters return back to the state of followers with rate α and retweeters remain as retweeters with rate η. At time t, external visitors spontaneously retweet the tweet at rate γ. The increase rate of external visitors is ω.

$$S(t+1) - S(t) = -\beta I(t)S(t) + \alpha I(t)$$

$$I(t+1) - I(t) = \beta I(t)S(t) + \eta I(t) + \gamma E(t) - \alpha I(t)$$

$$E(t+1) - E(t) = \omega I(t)$$

$$loss\,function = \sum_{t=1}^{w}[(S(t) - S^p(t))^2 + (I(t) - I^p(t))^2 + (E(t) - E^p(t))^2]$$

$S^p(t)$, $I^p(t)$ and $E^p(t)$ represent the predicted values for $S(t)$, $I(t)$ and $E(t)$. At the training process, the parameters are trained by back-propagation in order to minimize the loss function. The retweeting number can be estimated from the trained model (Wang, et al., 2013).

By extending the epidemic disease model, this method decreases the errors compared to some other methods (Linear Regression Support Vector Regression and etc.). This model considers both external (strangers) and internal (followers) factors. However, all retweeters at a time stamp are treated as a group. In other words, the algorithm has not considered that different individuals have different transmission rates β. The retweeter groups may change according to different tweets sent by the same twitter user. Content classification is also very important for predicting retweeting behaviors. It is also hard to choose the proper window size (time period) of training data to get the most accurate result. If the window size is too large, the tweets from a long time ago have huge impacts on the parameters and the parameters may not be able to estimate the latest state (amount of retweets). If the window size is too small, the model cannot be trained sufficiently as the importance of historical tweets has been ignored.

Retweeting Analysis based on User Interests Model

This approach involves content classification. It classifies user tweets into different categories by Bayes model. The algorithm predicts retweeting behaviors by measuring user interests on tweets of different categories. The probability of word w_i in a certain category (topic) C_i is calculated by:

$$m_{ij} = P(w_j|C_i) = \frac{N_i(w_j)+\delta}{\sum_{w_j \in WC_i} N_i(w_j)+\delta|WC_i|}$$

Where δ is a smoothing factor and $N_i(w_j)$ is the number of w_j in category C_i. $|WC_i|$ is the total number of words in category C_i.

With m_{ij}, the probability of a tweet t of topic C_i can be obtained by using Bayes theorem:

$$P(C_j|t_i) = \Pi_{w_k \in T_i} P(C_j|w_k)$$

$$P(C_j|w_k) = \frac{P(C_j)P(w_k|C_j)}{P(w_k)} = P(C_j)\frac{P(w_k|C_j)}{\sum_{C_i \in C} P(C_i)P(w_k|C_i)}$$

The whole algorithm works as follows:

Input: tweet category dataset G, a user u and all of his tweets $T = \{t_1{}^u, t_1{}^u, ..., t_n{}^u\}$, a tweet t from one of the users which u follows.

Output: Whether user u will retweet t.

Step 1: Compute tweet category feature matrix M.

Step 2: for i = 1 in n:

$$P(C_j|t_i^u) = \Pi_{w_k \in T_i} P(C_j|w_k)(1 \leq j \leq p)$$

$$P(C_q|t_i^u) = max(P(C_j|t_i^u))(1 \leq j \leq p)$$

$$C(t_i^u) = C_q$$

Step 3: for l = 1 in p:

$$T_l^u = \{t_r^u | C(t_r^u) = C_l\}$$

$$F(u, C_l) = \lambda_{c(l)}$$

$$\lambda_{C(l)} = \sum_{t_r^u \in T_l^u} P(C_l|t_r^u)/|T_l^u|$$

Step 4: User u will retweet t if $P(C(t)|t) \geq \lambda_{C(t)}$, otherwise u won't retweet it.

The author claimed that they can achieve a competitive result compared to other retweet behavior prediction algorithms. They explored how user interests would affect users' retweet behavior. However, this approach does not consider the relationship between users and the popularity of each user. In addition, they used a very simple probabilistic language model to classify the tweets' topic, and this may cause inaccurate classification since it assumes that each word occurrence is independent. This can be improved by using a recurrent neural network for topic analysis (Huang, et al., 2014).

Voting Algorithm

A simple voting algorithm model can be described as an iterated filtering process of mutually updating information for both candidates and voters.

$r -> l_i$ denotes that voter V_r votes for the candidate i on list l

For each V_r, his/her trustworthiness is denoted by T_r

ρ defines the score for candidate e.g. ρ_{li} defines the score for the candidate on list l row i

n_l denotes the number of candidates on a list l.

Initialization:

$$T_i^{(0)} = 1$$

$$\rho_{li}^{(0)} = \frac{\sum\limits_{r:r \rightarrow l_i} T_r^{(0)}}{\sqrt{\sum\limits_{1<=j<=nl} (\sum\limits_{r:r \rightarrow l_j} T_r^{(0)})^2}}$$

Repeat:

$$T_i^{(0)} = \rho_{l_i}^{(k)}$$

$$\rho_{li}^{(0)} = \frac{\sum\limits_{r:r \rightarrow l_i} (T_r^{(k+1)})^{\alpha}}{\sqrt{\sum\limits_{1<=j<=nl} (\sum\limits_{r:r \rightarrow l_j} (T_r^{(k+1)})^{\alpha})^2}}$$

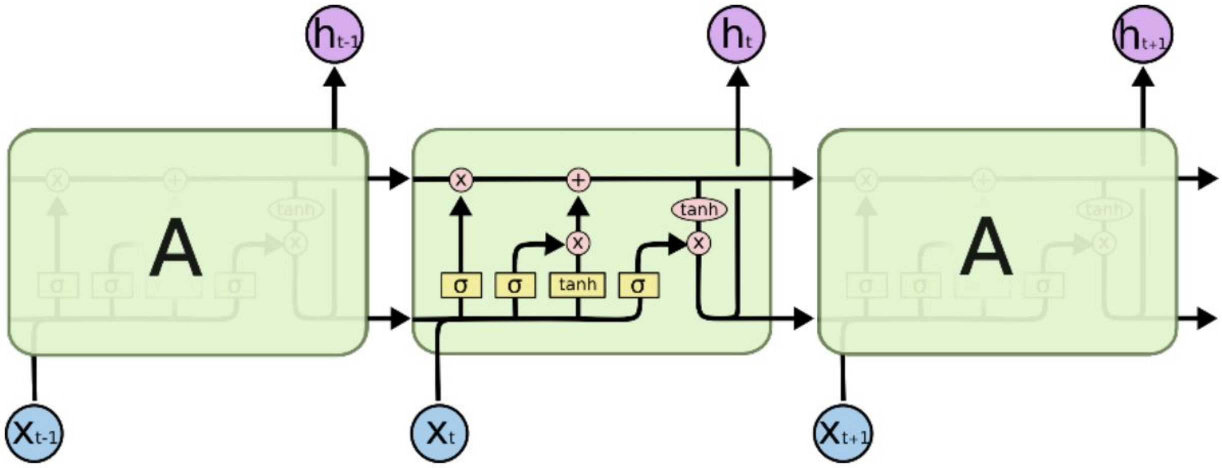
The algorithm will eventually converge as the trustworthiness of each voter and scores of candidates converge to constant values (Allahbakhsh and Ignjatovic, 2012).

Sentiment Analysis with Recurrent Neural Network

Recent progress in neural networks has made it outperforms other sentiment analysis methods such as probabilistic language model. The neural network is a computational model consisting of connected neurons and neurons are organized in layers. Different layers perform different transformations on their inputs. Signals travel from the first (input) to the last (output) layer. Training a network to do sentiment analysis consists of feeding the network with training example and label pairs, then comparing the output with the label of the corresponding example and updating the network weights by back-propagation.

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This allows it to exhibit dynamic temporal behavior which is the key to sentiment analysis.

In this project, we use a variant of a recurrent neural network called Long Short-Term Memory (LSTM). LSTMs are well suited to learn from experience when there are very long time lags of unknown size between important words, which makes them especially attractive for applications in sentiment analysis.



(**Fig.1** LSTM model. *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*)

LSTM model has a cell, an input gate, an output gate and a forget gate. The cell is used to "remembering" information over arbitrary time intervals. The forget gate decide how much information will be 'forgot' from the last time step, the input gate decide how much information will be transferred to the cell and the output gate generate the result for the current time step. At each time the output and memory cell will be parts of the input for the next time step. These gates regulate the flow of information that goes through the connections of the LSTM (Figure 1).

Gates:

$$\mathbf{f}_t = \sigma\left(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f\right)$$
$$\mathbf{i}_t = \sigma\left(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i\right)$$
$$\mathbf{g}_t = \tanh\left(W_g \mathbf{x}_t + U_g \mathbf{h}_{t-1} + \mathbf{b}_g\right)$$
$$\mathbf{o}_t = \sigma\left(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o\right)$$

State:

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i}_t \odot \mathbf{g}_t$$

Output:

$$\mathbf{h}_t = \tanh \mathbf{c}_t \odot \mathbf{o}_t$$

In sentiment analysis, RNN was fed with a tweet which is a sequence of words(embedding format) at each time step and the network will predict a topic of the fed tweet. Then comparing the target category with the predicted topic and alternate weights of the network to reduce the error by back-propagation (Sundermeyer, et al., 2012).

Solution

An iterated filtering and real-time updating algorithm is conducted to calculate the importance of a specific user and predict an approximate amount of retweets of a specific tweet. The proposed solution is based on three main factors: the strength of the relationship between users, aggregated message weights according to the importance of retweeters and weighted discount factor regarding the timeline of messages. The predicted number of a user's retweets is proportional to the user's importance at specific timestamp. The following paragraphs will introduce the approach step by step.

Basic Concepts and Notations

Assume that there has a set *V* that contains *N* twitter users $\{V_1, V_2, \cdots V_n\}$.

Importance (rank) of user (twitter account) V_i is denoted by S_i.

m_{ti} denotes the message (tweet) sent by user i at time stamp t.

w_{ti} is the weight of the message sent by user i at time stamp t.

Let η to denote weighted time discount factor.

$V_i \rightarrow m_{tj}$ represents the user i retweeted message sent by user j at time stamp t.

$\#m_{v_i}$ is the total number of direct messages that sent by user i.

$\#followee_i$ is the total number of users whose messages have been retweeted by user i.

Let $\#m_{v_i \rightarrow v_j}$ to represent the total number of messages that user i has retweeted by user j at time stamp t.

$R(u_{ij})$ is relationship function used to evaluate the strength of the relation between user i and user j.

$D(t)$ is a time-discounted function.

$Retweet(V_i)$ is the predicted retweet amount of user V_i will get in the next time stamp

$\#retweet_{t_j \sim t_k}$ is the total number of retweets between time j to k

Basic Twitter Prediction Algorithm (BTPA)

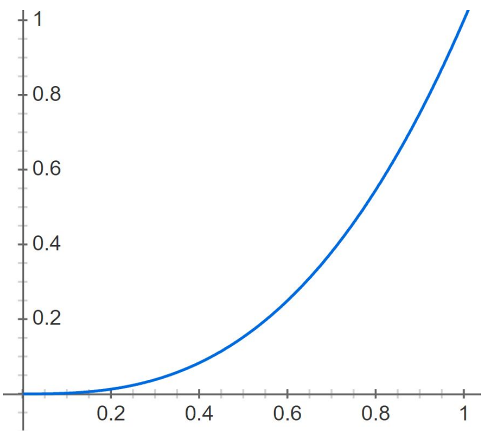
Relationship function $R(u_{ij})$ is determined by how frequently user i will retweet user j's twitter. The stronger the relationship, the more likely a tweet will be retweeted in the future. It is calculated based on average amount of retweets.

$$R(u_{ij}) \propto \frac{\#m_{v_i \to v_j}}{\#m_{v_i}}$$

In this scenario the $\frac{\#m_{v_i \to v_j}}{\#m_{v_i}} \in [0, 1]$. Consider user V_j is a celebrity (assume every tweet sent or retweeted by V_j will get more than 10000 retweets), assume V_j has a possibility of 0.5 to retweet messages of user V_i ($\frac{\#m_{v_i \to v_j}}{\#m_{v_i}} = 0.5$) and messages V_i will be only retweeted by V_j. It is biased to claim that a message sent by V_i will get 5000 retweets. x^e (Figure 2) will be used to punish the weak relationships between users.

Considering the external factor, $\frac{1}{|V|}$, represents the small possibility that a tweet can be retweeted by external visitors or self-retweeted.

$$R(u_{ij}) = \begin{cases} \left(\frac{\#m_{v_i \leftarrow v_j}}{\#m_{v_i}} + \frac{1}{|V|}\right)^e, & \text{if } \frac{\#m_{v_i \leftarrow v_j}}{\#m_{v_i}} + \frac{1}{|V|} < 1 \\ 1, & \text{otherwise} \end{cases}$$



(**Fig. 2** Plot of $f(x) = x^e$)

A message's weight, representing the impact of the message, is proportional to the importance of twitter users who have retweeted this message. As people with higher rankings retweet the message, the more likely these tweets will be seen and retweeted by more people. Such that these tweets should be assigned with higher marks for the influence they have. The weights are normalized to one at each time stamp. Relationships between users will be reflected in weights of messages.

$$w_{ti} = \sum_{j:V_j \to m_{ti}} \frac{S_j R(u_{ji})}{\sum_{c}^{N} \sum_{k:v_k \to m_{ti}} S_k R(u_{ki})}$$

Since the “audience” of a twitter user will change as time goes by, recent messages will be more reliable than earlier messages in the analysis. A time discount function $D(t)$ is used to decrease the contribution of the weight of earlier tweets to users’ rankings in the user set.

$$D(t) = \eta^t,\ \eta = 0.9$$

Importance of user V_i is proportional to all his/her past tweets’ weights multiplied by time discount function.

$$S_i = \frac{\sum_{t:m_{ti}} w_{ti} D(t)}{\#m_{v_i} \#followee_i}$$

Normalize S_i:

$$S_i' = \frac{S_i}{\| S_i \|}$$

After the message weights and users’ importance converge, the rankings of users can be considered as the contribution of users to the total amount of retweets at a time interval. The predicted retweet count of user V_i can be calculated as follows:

$$Retweet(V_i) = \#retweet_{t_j \sim t_k} S_i$$

Variation Twitter Prediction Algorithm (VTPA)

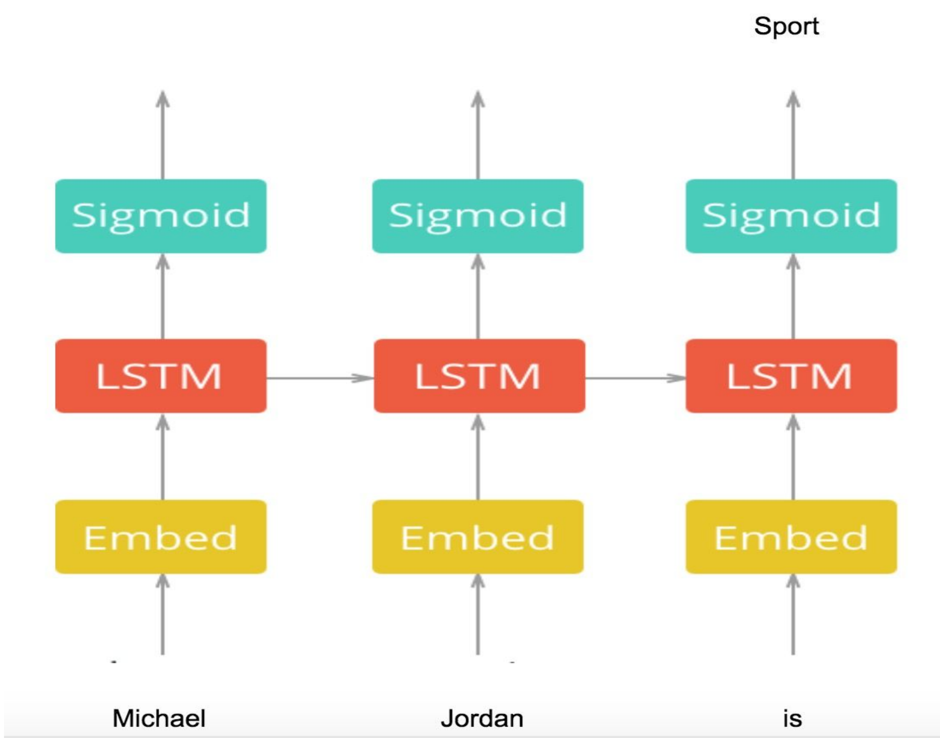
Since the contents of tweets will also have impacts on the number of retweets, the algorithm can be modified to take account the semantic meaning of tweets by using RNN (LSTM).

Word processing

To convert a tweet to an input vector of our model we need to process raw data to get a sequence of words in vector representation as the inputs to the network. In this project, instead of training word embeddings from scratch, we use the GloVe embeddings from Stanford NLP group (Pennington, et al., 2014).

Long Short Term Memory

Recall the relationship function $R(u_{ij})$ is determined by how frequently user i will retweet user j's twitter and it is a positive value. In the variation of the algorithm, it becomes a vector $R_{u_{ij}}$. The c^{th} item of it, $R_{cu_{ij}}$ can be considered as for how frequently user i will retweet user j's twitter in category c.



(**Fig. 3** Simplified workflow of RNN.)

Recurrent network (RNN) with long short term memory (LSTM) is used for sentiment analysis in this project. Most tweets contain a sequence of words. The sequence of words and the order of words are the most important factors for categorizing tweets. Using LSTM, not only the information of every single word can be extracted, but also from the order of words (unlike previous simple probabilistic language model). Firstly each word of tweets will be converted to word embeddings (vector representation). The training process is completed by supervised learning, we feed RNN (Figure 3) with a target topic and a tweet which is a sequence of the vector, then the network will compute different values for each topic. We use softmax function to turn linear values o_j which is the output value of topic j into a probability distribution estimating the probability of the tweet belongs to topic j .

$$prob(topic_j|tweet) = \frac{exp(o_j)}{\sum_{j'=1}^{V} exp(o_{j'})}$$

We then seek to maximize the probability with back-propagation

$$prob(topic_j = targetTopic|tweet) = \frac{exp(o_j)}{\sum_{j'=1}^{V} exp(o_{j'})}$$

Training Details

The exact architecture used in this project is as follows. The input to the neural network consists of a sequence of words in vector representation from GloVe embedding with size 200. The hidden layer size is 100 and applies a rectifier nonlinearity. The final layer is a fully-connected layer leading into a soft-max classifier consists of 8 outputs corresponding to each topic. For the hyper-parameters, we use Adam Optimizer with exponential decay learning rate starts from 0.1 and step size 1000. The batch size is 128.

After classifying all the tweets in the dataset, we separate the datasets into 8 subsets based on their topics, namely, technology, politics, life, sports, entertainment, health, travel, finance. Then the relation matrix $R_{cu_{ij}}$ can be calculated by:

$$R_{cu_{ij}} = \begin{cases} (\frac{\#m_{cv_i \leftarrow v_j}}{\#m_{cv_i}} + \frac{1}{|V|})^e, & \text{if } \frac{\#m_{cv_i \leftarrow v_j}}{\#m_{cv_i}} + \frac{1}{|V|} < 1 \\ 1, & \text{otherwise} \end{cases}$$

Where $\#m_{cv_i}$ represents the total amount of tweets of category c sent by user V_i. $\#m_{cv_i \to v_j}$ to notate the number of times that user V_i has been retweeted V_j in tweet category c.

8 separate models will be trained on these subsets until converging.

Testing

In the test time, we first classify the tweet into the topic with the highest probability. Then use the corresponding rank vector S_c to compute the predicted number of retweets by

$$Retweet(V_i) = \#retweet_{t_j \sim t_k}(c) S_c(V_i)$$

Converge Proof

In order to prove convergence of the algorithm. The algorithm will be rearranged in the matrix form.

For each timestamp t, define a $V \times V$ matrix M^t to recall the retweeting behavior between users. If M^t_{ij} is equal to 1, it means user i's tweet is retweeted by user j at timestamp t, and a small positive number ε means there is no retweet between user i and user j.

The relation function $R(u)$ also can be expressed as a $V \times V$ matrix R with all positive entries. Each entry R_{ij} represents the possibility of the message sent by user i will be retweeted by user j. In VTPA, it represents the possibility of the message in a specific category sent by user i will be retweeted by user j.

W^t_k is the matrix of all weights of tweets sent at t^{th} time stamp at k^{th} iteration .

S_k is the score vector of all twitter users at k^{th} iteration.

T is the total number of timestamps, η is the time discount factor, $0 < \eta \leq 1$.

$$W^t{}_k = M^t \otimes R S_k \eta^{T-t} \qquad (1)$$

At $(K+1)^{th}$ iteration, the score vector S_{k+1} can be expressed by

$$S_{k+1} = \sum_{t=0}^{T} W^t{}_k \qquad (2)$$

Substitute (1) to (2):

$$S_{k+1} = \left(\sum_{t=0}^{T} M^t \otimes R \eta^{T-t}\right) S'_k \quad (3)$$

Normalize (3)

$$S'_{k+1} = \frac{S_{k+1}}{\| S_{k+1} \|} \qquad (4)$$

Let:

$$G = \sum_{t=0}^{T} M^t \otimes R \eta^{T-t} \qquad (5)$$

Substitute (5) to (3):

$$S_{k+1} = G S'_k \qquad (6)$$

Substitute (6) to (4):

$$S'_{k+1} = \frac{G S'_k}{\| G S'_k \|}$$

This can be unrolled to $S_{k+1} = G^{k+1} S_0$. Since S can be written as a linear combination of the eigenvectors of G:

$$S = c_1 \lambda_1 + c_2 \lambda_2 + c_3 \lambda_3 + \ldots + c_n x_n$$

Then

$$S_{k+1} = G^{k+1}S_0 = \sum_{i=1}^{n} c_i G^{k+1} s_i = \sum_{i=1}^{n} c_i \lambda_i^{k+1} s_i$$

Since G is a matrix with all positive entries(z_k and η are positive, M^t and R are all positive real matrix), according to the Perron-Frobenius theorem:

A real square matrix with positive entries has a unique largest real eigenvalue and that the corresponding eigenvector can be chosen to have strictly positive components.

We can define that

$$\lambda_1 > \lambda_2 >= \lambda_3 >= \ldots >= \lambda_n$$

This implies that S_k will converge to a unit vector which is a scalar multiple of the eigenvector x_1 corresponding to λ_1 with some tolerance. This is the power iteration method.

Preliminary Developments

Input Data

Data Collection

We created a python script using python-twitter API (reference) to collect the twitter information.

Because of rate limitation of python-twitter API and the storage space limitation, it is unrealistic to track each twitter account. Thus, we divide twitter users into two categories, which are celebrities (average retweets amount larger than 200) and common users (average retweets amount equal or smaller than 200). The relevant information including average retweets amount of a user and messages' retweeter ID lists has been stored in the database. Furthermore, ten twitter accounts have been used to increase the API call rate.

The script is able to continue tracking the tweets update of each user belongs to celebrity as well as finding the important retweeters of a specific tweet. After an important retweeter account is found, this account will be stored to the celebrity list and tracked at the next iteration. Common users won't be tracked because their public effects can be ignored (small contribution to messages' retweet amount). The pseudocode of the script as follows:

```
while True:
    for celebrity in celebrity_list:
        tweet_Id ← get the recent tweet ID of the celebrity
        if the tweet_Id not exist in database:
            create a record with current tweet_Id in database
        recent_retweeters_list ← get recent retweeters of the tweet_Id
        store the new retweeters to the corresponding record
        for retweeter_Id in recent_reteeters_list:
            if retweeter_Id not exist in database:
                ave_retweets ← get average retweet amount of this retweeter_Id
                if ave_retweets > 200:
                    store retweeters information as celebrity
                    add retweeter to celebrity_list
                else:
                    store retweeters information as common user
```

The python script has been deployed on Google Virtual Machine and kept updating 24 by 7. After 2 months mining, there has been totally over 1.8 million common users, one thousand celebrities and 12000 tweets sent by them have been retrieved.

Data Process

After the data has been collected, it was processed before running the algorithm. A three-dimensional matrix is used for extraction of data. As each row represents a timestamp t and each column represents a single user i, the retweet list is stored in each matrix slot. e.g. Matrix[t][i] stores the list of retweeters who have retweeted the tweets sent by user i at time stamp t. Each timestamp is defined as a daily interval. All the retweeters were aggregated based on daily interval.

The original time stamps are modified to integers starting from zero for the convenience of matrix representation. The original twitter IDs are also reorganized to integers starting from zero.

$$
\begin{array}{cc}
 & \text{Users} \\
\text{Timestamp} & \begin{bmatrix} [1,7,4] & \dots & [-1] & \dots & [1,3,5] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [12,18] & \dots & [0] & \dots & [\,] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [-1] & \dots & [\,] & \dots & [6,2,4] \end{bmatrix}
\end{array}
$$

In the matrix, the empty slot represents the that the user did not send any tweets at timestamp t and -1 means that users did not get any retweets for that tweet.

This matrix is used for the input data of algorithm training.

Algorithm Implementation

BTPA is implemented in Python 3.6 based on the basic version. At the training process, new messages can be added to the input matrix at each new timestep. The users' scores and message weights are updated synchronously. The structure of training process is like following:

$score \leftarrow$ initalize all users' scores to 1
$R \leftarrow$ calculate relation matrix R
$\eta \leftarrow 0.9$
$\alpha \leftarrow$ threshold for stop training
while True:
    $prev\_scores \leftarrow scores$
    $message\_weight \leftarrow$ update $message\_weight$ by $scores$ and matrix R
    $scores \leftarrow$ update $scores$ by $message\_weight$ and η
    $error \leftarrow$ square distance between $prev\_scores$ and $scores$
    if $error < \alpha$:
        break

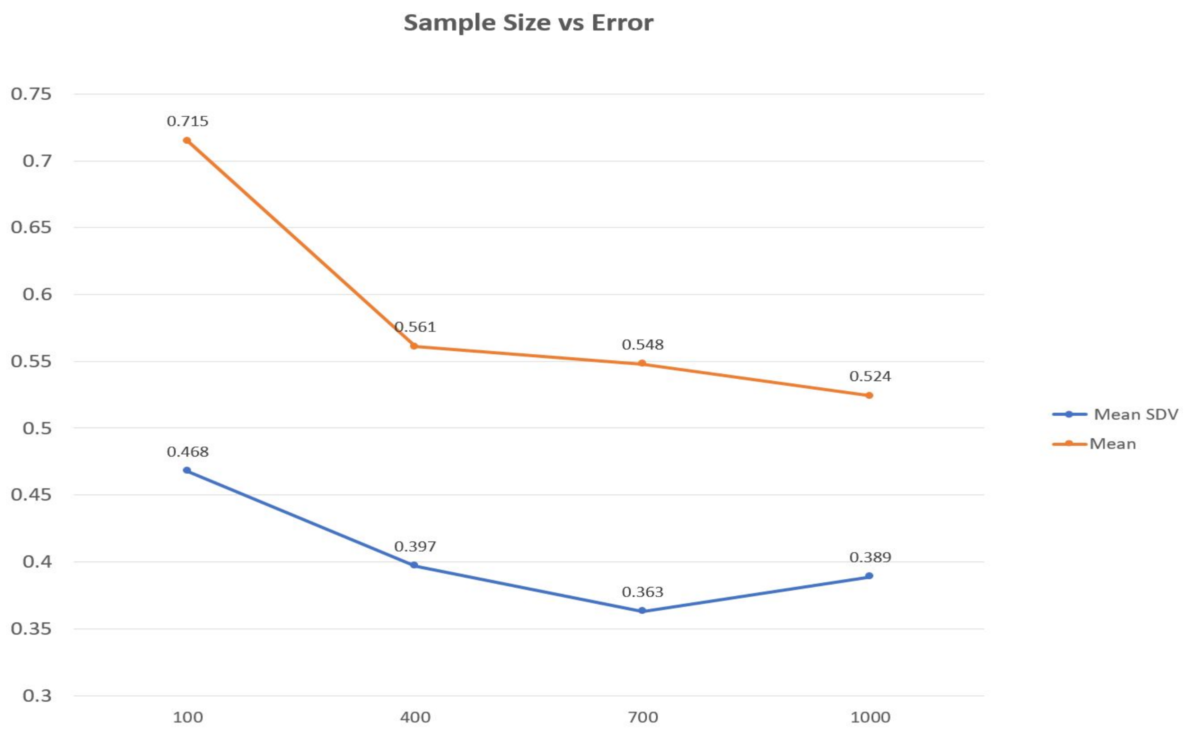
Evaluation of BTPA

We have conducted two sets of tests: the first is to determine the effect of the sample sizes and the second is to determine the effect of the training interval.

Errors by Choosing Different Sample Sizes

As discussed previously, we have collected data from 1000 twitter users about their retweeting details over 54-day duration. There were in total 12000 tweets sent by them. The algorithm is evaluated by its errors. The errors are analyzed by the percentage of the mean of errors ($\frac{Predicted\ Value - Actual\ Value}{Actual\ Value}$). We use the standard deviation of the error to evaluate how well is the algorithm adapted to different situations.

The first experiment is to measure the effect of the sample size (size of users). The experiment was conducted on four sample sizes separately: 100, 400, 700 1000. The samples are randomly chosen from 1000 tracked users. The experiment uses first 21 time intervals (days) as a training set and 10 tweets in same time interval were used as the test set. The mean of errors and standard deviation of mean of errors are given as following (Figure 4):

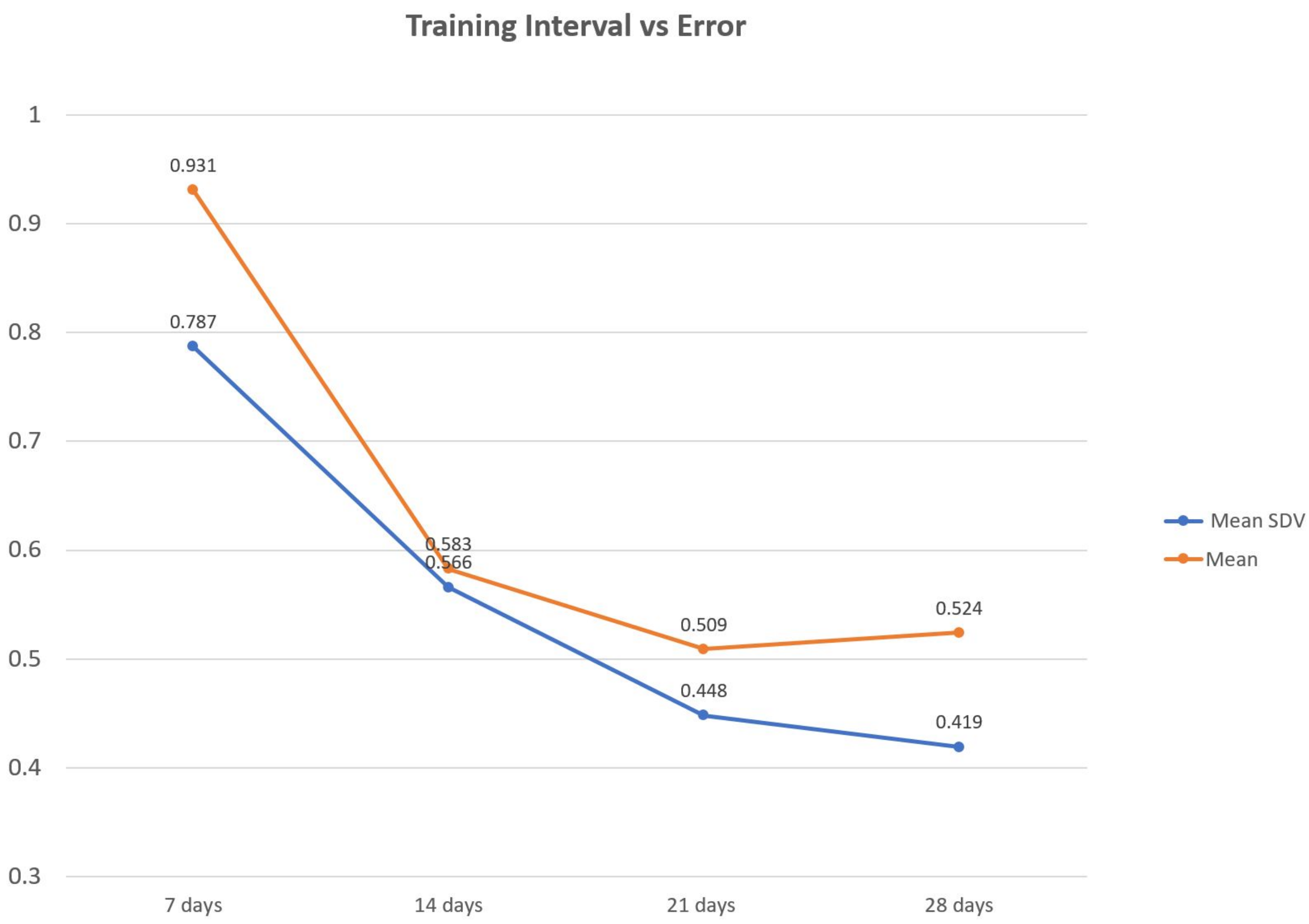


(**Fig. 4** Prediction errors on different sample sizes)

From the first experiment conducted, as sample size increases, the mean of errors has the decreasing trend. There is a significant decrease in testing error from sample size 100 to 400. With small sample size, like 100, the amount of messages and retweets is limited and predictions are not reliable. After training with the sufficient sample size (400 to 1000), the error is stabilized at approximately 0.52.

Errors in Different Training Time Interval

The second experiment is to measure the effect of the training interval on 1000 sample sizes (which gave us the best result from the first test). The experiment was conducted at four training intervals separately: 7-day, 14-day, 21-day, 28-day and 10 tweets in same time interval were used as the test set. The mean of errors and standard deviation of the mean of errors are given as following (Figure 5):



(**Fig. 5** Prediction errors on different training interval)

From the second experiment conducted, as training time interval increases, the mean of errors has the decreasing trend. There exists significant decrease of discrepancies from training interval of 7-day to 21-day. It reflects that the BTPA can take the impact of historical tweets into consideration in a proper way. In addition, the prediction errors from 21-day to 28-day are approximately same. The historical tweets will not be over-estimated and it proves that our algorithm will not be over-fitting when training with large training time intervals.

Conclusions

In order to predict information diffusion on social networks, two feasible solutions based on iterated filtering algorithms and recurrent networks of twitter retweeting predictions have been carried out after summarizing advantages and disadvantages of current methods and investigations on computing resources and development weapons. A prototype of BTPA also has been implemented, and preliminary experiments have already demonstrated its considerable performances. We expected that VTPA will further improve the accuracy of prediction. The implementation of VTPA and careful evaluation will be carried subsequently in the near future.

Bibliography

Allahbakhsh, M. and Ignjatovic, A., 2012. Rating through voting: An iterative method for robust rating. *arXiv preprint arXiv:1211.0390*.

Hong, L., Dan, O. and Davison, B.D., 2011, March. Predicting popular messages in twitter. In *Proceedings of the 20th international conference companion on World wide web* (pp. 57-58). ACM.

Huang, D., Zhou, J., Mu, D. and Yang, F., 2014, December. Retweet behavior prediction in twitter. In *Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on* (Vol. 2, pp. 30-33). IEEE.

Pennington, J., Socher, R. and Manning, C., 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Sundermeyer, M., Schlüter, R. and Ney, H., 2012. LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Tang, X., Miao, Q., Quan, Y., Tang, J. and Deng, K., 2015. Predicting individual retweet behavior by user similarity: A multi-task learning approach. *Knowledge-Based Systems*, *89*, pp.681-688.

Wang, H., Li, Y., Feng, Z. and Feng, L., 2013. ReTweeting analysis and prediction in microblogs: An epidemic inspired approach. *China Communications*, *10*(3), pp.13-24.

Zaman, T., Fox, E.B. and Bradlow, E.T., 2014. A Bayesian approach for predicting the popularity of tweets. *The Annals of Applied Statistics*, *8*(3), pp.1583-1611.