9-14-2018

# A Framework for Detecting Injected Influence Attacks on Microblog Websites Using Change Detection Techniques

Vishnu S. Pendyala

Yuhong Liu
*Santa Clara University*, yhliu@scu.edu

Silvia M. Figueira
*Santa Clara University*, sfigueira@scu.edu



## Recommended Citation

Pendyala, V. S., Liu, Y., & Figueira, S. M. (2018). A framework for detecting injected influence attacks on microblog websites using change detection techniques. Development Engineering, 3, 218–233. https://doi.org/10.1016/j.deveng.2018.08.002

Development Engineering 3 (2018) 218–233

Contents lists available at ScienceDirect

Development Engineering

journal homepage: www.elsevier.com/locate/deveng

# A framework for detecting injected influence attacks on microblog websites using change detection techniques

Vishnu S. Pendyala*, Yuhong Liu, Silvia M. Figueira

*Santa Clara University, USA*

ARTICLE INFO

*Keywords:*
Cumulative sum
Discrete Kalman Filter
Sentiment analysis
Online Social Networks
Twitter bot
Attack injection
Microblogging

ABSTRACT

Presidential elections can impact world peace, global economics, and overall well-being. Recent news indicates that fraud on the Web has played a substantial role in elections, particularly in developing countries in South America and the public discourse, in general. To protect the trustworthiness of the Web, in this paper, we present a novel framework using statistical techniques to help detect veiled Web fraud attacks in Online Social Networks (OSN). Specific examples are used to demonstrate how some statistical techniques, such as the Kalman Filter and the modified CUSUM, can be applied to detect various attack scenarios. A hybrid data set, consisting of both real user tweets collected from Twitter and simulated fake tweets is constructed for testing purposes. The efficacy of the proposed framework has been verified by computing metrics, such as Precision, Recall, and Area Under the ROC curve. The algorithms achieved up to 99.9% accuracy in some scenarios and are over 80% accurate for most of the other scenarios.

## 1. Introduction

Social media has played an important role in Presidential elections in the United States as far back as 2008, during Barack Obama's election campaign (Cogburn and Espinoza-Vasquez, 2011). Its influence is so powerful that the political cyber hacker, Andres Sepulveda once said (Laquintano and Vee, 2017), “When I realized that people believe what the Internet says more than reality, I discovered that I had the power to make people believe almost anything.” His conviction turned out to be disastrous to the developing nations in Latin America. He proved he was right by faking social media accounts and using them to fabricate trends to sway the results of various Presidential elections in South American countries. This type of attack is also named as injected influence attack, which can be defined as the activity of posting malicious microblogs, often but not always, using automated means in order to hijack the opinions of the other users. The project described in this paper is an effort to detect and prevent such malicious attacks in the future.

Studies such as (Pak and Paroubek, 2010) have shown that Twitter corpus adheres to the Zipf law and can be used for opinion mining using sentiment analysis. Therefore the corpus of microblogs can be modeled as a Zipfian distribution. A Zipfian distribution is a type of discrete power law probability distribution. In an extensive survey on opinion mining and sentiment analysis (Liu, 2012), Liu discussed the statistical characteristics of the sentiment scores of opinion corpora.

Most of the current approaches, such as (Vosoughi, 2015), treat the problem of detection of rumors on Twitter as a classification task and use machine learning algorithms, such as SVM and Naive Bayes to train the classifier. The existing work, however, fails to sufficiently exploit the underlying characteristic of the problem, which is the fluctuation in the opinions expressed in the microblogs. In addition, as smart attackers often mimic normal users' behavior patterns to prevent themselves from being detected, machine learning algorithms which focus on specific patterns may often not perform well. For instance, the solutions presented in Vosoughi (2015) were successful in identifying around 70% rumors correctly, as compared to the results from the solution presented in this paper.

In this work, we propose to detect anomaly from another angle - changes based on the hypothesis that to influence the public opinions, the fake microblogs generated by malicious attackers will inevitably cause changes in the normal opinions. The framework we propose in this paper using change detection techniques utilizes this underlying characteristic to accomplish the task of discerning the fake tweets from the real ones, which is expected to result in better accuracy than the routine Machine Learning approaches.

Please note that occasionally, normal users' sentiments may be shifted due to the release of some startling news. However, such shifts can be easily validated by cross-checking other information sources,

* Corresponding author.
*E-mail addresses:* vpendyala@scu.edu (V.S. Pendyala), yhliu@scu.edu (Y. Liu), sfigueira@scu.edu (S.M. Figueira).

https://doi.org/10.1016/j.deveng.2018.08.002
Received 20 February 2018; Received in revised form 19 August 2018; Accepted 19 August 2018
Available online 29 August 2018

such as recent released news on other media. For brevity and focus, we do not include the cross-checking in this work and leave it as a future direction. The experimental setup for this paper is designed such that common injected attack scenarios are covered.

In addition, to cover the few cases where opinion shifts caused by injected attacks cannot be distinguished from those caused by genuine opinion shifts, the framework presented in this paper can be further strengthened by correlating and corroborating the results with those obtained by other techniques described in related literature. This paper is hoped to initiate a substantial discussion on solving an important problem that today's democracies, such as in the developing countries in South America, as highlighted earlier and the world in general is facing.

Cumulative Sum (i.e. CUSUM) is a statistical analysis technique (Page, 1954) used for change detection. The technique was successfully used on numerical data for anomaly detection in feedback-based online reputation systems (Liu and Sun, 2010). A Kalman Filter is typically used to predict values of data using a recursive algorithm. A series of observed values differing substantially from predicted values can indicate change.

Given that the sentiment scores from a microblog corpora can be modeled as a probability distribution, we hypothesize that statistical change detection techniques using CUSUM, modified for our purpose and a Kalman Filter can be applied to the sentiment scores of a microblog corpora to identify opinion shifts and therefore hacker attacks. Sentiment scores of microblogs on any given topic can be expected to be reasonably predictable. Delirious posts are commonly identified by humans, based on their out-of-whack sentiment. Machines can achieve the same result once the sentiment is quantified as a numerical score. Change detection techniques help in this process.

For this work, we chose the 2009 Iran elections as the domain, for reasons discussed in one of the following sections. We scored the tweets on the topic using automated sentiment analysis. We then applied our modified CUSUM (mCUSUM) and Discrete Kalman Filter algorithms to study the fluctuations in the sentiments. There was no apparent adverse impact noticed to the extent of suspecting an attack. We then injected a number of tweets with negative sentiment to simulate several scenarios. We repeatedly applied the two techniques to analyze the tweet sentiments in the various scenarios and were successful in detecting the injected tweets with an impressive accuracy of around 90%. For simplicity, we use the words, microblogs and tweets interchangeably, but the techniques discussed in this paper are by no means restricted for use with Twitter.

To the best of our knowledge and literature survey, the paper is unique in efficiently and successfully proposing a framework to use Change Detection techniques such as modified CUSUM (mCUSUM) and Kalman Filter with sentiment analysis to detect OSN hacker attacks intended to influence voters in microblogs. Specifically, we have modified the basic CUSUM in a novel way to make it better fit our application scenario. We tested both the mCUSUM and Kalman Filter Change Detection techniques, applied each one to various scenarios, and compared the results. The results confirmed our hypothesis, so the techniques explored here can possibly be extended to solve similar problems after quantifying the sentiment trends or other critical aspects of information. Moreover, based on the testing results, we further propose a comprehensive way to perform anomaly detection by integrating mCUSUM and Kalman Filter in a flexible way. The model we use is generic enough to be implemented as a framework.

Last but not least, due to the wide adoption of Web information, protecting a trustworthy Web is also essential for other domains. For example, there are a number of humanitarian projects that have been made possible by the Web. One such possible application is Web-based medical diagnosis using the techniques presented in Pendyala et al. (2014) and Pendyala and Figueira (2017). These rely heavily on the truthfulness of the underlying data, which is not entirely tamper-proof. The use of CUSUM, modified for our purposes, and Discrete Kalman Filtering techniques presented in this paper for detecting hacker attacks is hoped to pave way for detection of tampering of critical data such as in the medical domain as well.

The rest of the paper is organized as follows. Section 2 presents existing literature. The next section, section 3 discusses the design aspects, detailing the framework, the approach, the techniques, and the algorithms. Section 4 provides the experiment details and results. Section 5 concludes the paper with a discussion on the results and future directions.

## 2. Related work

There are a number of papers on the topics related to the influence of OSNs, anomaly detection, and misinformation containment areas, which are closely related to our project. We list some of the interesting ones in the following subsections.

### 2.1. Influence of social media

The first set of papers we examined relate to whether microblogs make a difference to the outcome of public opinion and decision making. To evaluate the influence of the tweets, we need a quantitative measure of their sentiment. Choy et al. did a sentiment analysis of the tweets related to the Singapore Presidential elections (Choy et al., 2011) to estimate the number of votes each candidate will get. Their work proves the important role that tweets play in elections to the high office and proves the correlation between the sentiment analysis of the tweets and the election results. Similarly, the authors of Tumasjan et al. (2010) examine the role tweets played in the German elections. One of their conclusions is that even mentioning the party name in the tweet has non-trivial impact. The more the number of mentions, the higher the chances of winning the elections.

In an extensive analysis of the Twitter corpus on two significant topics in the recent past, “Brexit” and “Trump”, Hall et al. (2018) lead us to a powerful conclusion, “Society might well need to quickly determine new ethical boundaries around the use of social media data analysis during election campaigns, or AI could determine who our next leaders will be.” Using analytical methods such as Sentiment Analysis, Temporal Profiling, LDA Topic Modeling, and visualization artifacts, such as Network Structure, they scrutinize the role that social media played in the two important referendums: Brexit and US Presidential elections.

Using quantitative analysis, authors of Jin et al. (2014a) portray how rumors spread on Twitter during the Ebola crisis in Africa, highlighting the impact lies on microblogging websites have had on developing countries. More literature survey shows that there is predominant evidence that there is substantial impact of the social media posts on the public opinion, supporting the purpose of this paper, which is to detect malicious use of microblogging during crucial events like elections. Burns and Eltham in their highly cited work (Burns and Eltham, 2009), examine the impact of Twitter on the Iran Election crisis that provides a sociological prelude to the technical discussion in this paper.

### 2.2. Anomaly detection

Other researchers have approached the topic of anomaly detection. The discussion on detecting certain type of security breach events such as “Sarah Palin's email account was hacked” from the tweets using semi-supervised learning methods in Ritter et al. (2015) gives good insights into the process of examining and making sense of odd-sounding tweets. Singh et al. (2014) propose ways to identify malicious users using five different classifiers and compare the results. They conclude that Random Forest resulted in highest accuracy. To improve the accuracy of our prediction, we may consider using Random Forest to extend our work in the future to confirm that the attacks we detect in

this work as coming from a malicious user.

Starbird et al. (2014) evaluate the effectiveness of the premise that crowdsourced information flows can contain the misinformation. Systematically analyzing the flow of tweets related to three rumors, they confirm that the premise is not valid and that there is a need for automated ways to detect falsity, which is what our paper presents. The approach taken in Castillo et al. (2011), Qazvinian et al. (2011), Kwon et al. (2017) and Yang et al. (2012) is somewhat similar to the one in our earlier work, (Pendyala and Figueira, 2015). The authors extract features from the tweets and build classifiers from them. The difference is that we used automated annotation and a different feature set in our previous paper.

In Jin et al. (2013, 2014b), the authors use Epidemiological modeling, specifically, the SEIZ model framework, to capture the diffusion of information on Twitter and suggest that the modeling can help in identifying rumors from facts. This can be used as an ensemble approach to what we present in this paper to improve the accuracy of detection. Authors of Chew and Eysenbach (2010) try sentiment analysis of the tweets and run Chi-square tests to examine the trends.

Hernandez-Suarez et al. (2018) apply Machine Learning to address the problem of predicting cyber-attacks on Twitter using sentiment analysis and L1 regularized regression model. They choose the tweets pertaining to Donald Trump's election as the President of USA and apply three Machine Learning Classifiers: Support Vector Machine, Maximum Entropy, and Naive Bayes. From the results, they conclude that Maximum Entropy performs best and use L1 regularization to improve the prediction.

Authors of Hamidian and Diab (2016) analyze Twitter users' belief in their posts using SVM Tree kernel model and other techniques to detect rumors. NLP techniques are used to determine the belief. Their premise, however may not be valid when the users are fake and malicious, a scenario that our work focuses on. A similar and more sophisticated approach is taken by the authors of Liu et al. (2015) to debunk rumors in real time. They use beliefs of the crowd and several other language features for the purpose. None of these papers consider the propagation patterns of the tweeted rumors, which is the crux of Wu et al. (2015). They use a hybrid SVM classifier on the propagation structure and some other features of the posts to detect rumors in the dataset they selected from Weibo. Their work can also be used in conjunction with our work to improve accuracy of the prediction.

### 2.3. Misinformation containment (MC)

There are a few studies on misinformation containment (MC), which aim to control the propagation of false information. An interestingly unique approach for Misinformation Containment (MC) is taken in Budak et al. (2011) and Nguyen et al. (2012). The authors try to identify Social Media influencers who can counter-campaign to contain the rumors. The authors of Budak et al. (2011) further prove that the problem is NP-hard and come up with a greedy algorithm and heuristics to solve the problem - a possible strategy that can be tried after detecting the misinformation using our work.

Cognitive Psychology, Page Rank algorithm, a “Retweet Graph” and other artifacts are used to design a framework for MC in Kumar and Geethakumari (2014). The framework can be used to complement our work in detecting the misinformation. Authors of Jain et al. (2015) use sentiment analysis like we do to detect rumors in their prototype, which they call as “The Twitter Grapevine”. Their approach is much simpler, in that they go by the “mismatch ratio” of the sentiment scores of tweets made by established news media to those of the general public and claim that it is effective enough. The role of rumors on the social media is much more profound than is captured in the mismatch ratio - a fact that the paper misses out.

Takahashi and Igata (2012) run content analysis and document classification techniques on the tweet to distinguish rumors from facts. This technique can be used on top of our work to enhance the accuracy. Statistical properties of viral misinformation are presented in Bessi (2017) using Extreme Value Theory methods. The dataset used is from Facebook posts. As future work, some of the properties presented can probably be leveraged in conjunction with our work to exploit temporal correlations.

As far as we can see, none of the existing literature surveyed used CUSUM, Kalman Filter, modified or directly, or any related techniques to approach the problem of injected attacks. We modified these algorithms in novel ways for the specific purpose, as described in the paper. The modified algorithms already resulted in high rates of accuracy, that there was really no need to come up with new ones or revise the techniques beyond what was done for this work.

## 3. Methodology

We formulate the problem as follows: Given a set of microblogs, $M$, on a topic, over a period of time, $T$, what is the probability, $p$, of detecting a deliberate and possibly hacked attack, $A$, to inject negative influence, $IFA$, on the Online Social Network (OSN), $N$, hosting the microblogs? Negative influence can be defined as the sentiment in bad thoughts about someone or something that encourages more bad thoughts.

It is possible that negative influence is not only caused by a hacked attack, but also by genuine happenings. To test the effectiveness of the algorithms in detecting hacked attacks, the experiments are designed in ways so as to simulate various scenarios of hacked attacks. The several scenarios used in the experiments described in this paper closely follow common attack patterns, which are not usually characteristic of genuine opinion shifts.

There may still be rare cases where genuine opinion shifts resemble those arising from common attack patterns. To differentiate between the two, the results from the framework presented in this paper can be cross-checked by other techniques covered in the related literature and also corroborated with more reliable sources such as professional news websites.

The solution needs to be general enough to be repeatable and consistent as in a framework, with pluggable components. Since we need to come up with a numerical value for the probability $p$, there is a need for quantifying the tweets in terms of what they represent, usually the sentiment they convey. The numerical values can then be processed by statistical anomaly detection tools. The framework is detailed in the following paragraphs.

### 3.1. Proposed anomaly detection framework

The approach presented here is general enough to use any Change Detection technique with any quantifying process such as sentiment analysis. The proposed framework for detecting attacks post-fact is illustrated in Fig. 1. The framework comprises of four pluggable modules detailed in the subsections below. We start by collecting microblogs from OSN $N$ on important topics that can potentially be targets of getting hijacked. Microblogging services such as Twitter provide streaming API's to search on specific hashtags and keywords for achieving this. Once the set of microblogs $M$ on a topic $T$ is collected over a period of time using the streaming API, the opinions are quantified using appropriate quantifying techniques such as sentiment analysis. The sentiment scores obtained for the microblogs are then passed over to statistical anomaly detection tools such as mCUSUM or Kalman Filter. The anomaly detection tool can then determine which of the microblogs in the given corpus could potentially be the result of a hack.

### 3.2. Corpora aggregation

Most microblog websites provide streaming APIs to access their worldwide data stream. Software using these streaming APIs run as

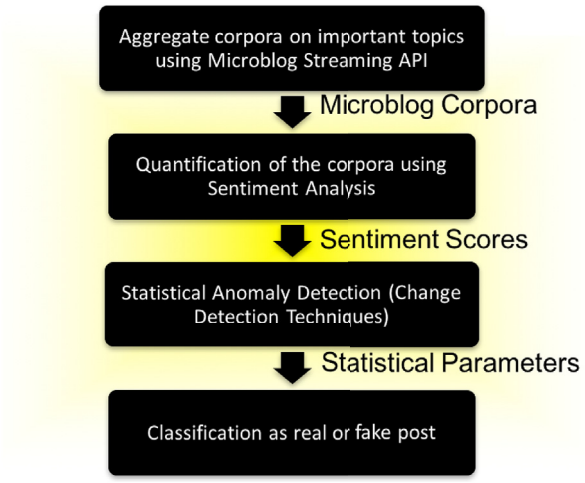


**Fig. 1.** Framework for offline detection.

clients to which user posts, events, and other data from the Microblog websites are automatically pushed without needing to poll the server. This data can be retrieved based on keyword and hashtag matches. The websites also provide REST APIs to get static information such user profiles or singular searches. Using a combination of the APIs, microblogs can be collected and aggregated into corpora based on topics. A few researchers have used these APIs and posted the resulting corpora online for other researchers' use. For this work, we use one such corpus as described in the subsequent sections.

### 3.3. Quantification of the microblogs: sentiment analysis

To facilitate analysis, the textual microblogs need to be converted to numbers representing the sentiment they convey. Sentiment Analysis helps us quantify the microblogs. Each microblog is assigned a sentiment score using NLTK python package, Affective Lexicon (AFINN) (Nielsen, 2011) and Vader Sentiment Lexicon (Hutto and Gilbert, 2014). We uniquely combined the latter two for better coverage. We downloaded the AFINN lexicon[1] and fused it with the other in Hutto and Gilbert (2014). Most of the words in the corpus we collected were in the lexicon. Words not in the lexicon do no affect the sentiment either ways. To improve the performance of the change detection, we tried a few schemes to adjust the granularity of the sentiment scores and chose the one which is best suited. In the final scheme, sentiment score is calculated as follows.

$$X_s = \sum \frac{X_i}{\sqrt{N}} \tag{1}$$

where $X_s$ is the sentiment score of the microblog, $X_i$ is the score of the individual word obtained from the AFINN and Vader lexicons mentioned earlier. $N$ is the number of words in the microblog.

### 3.4. Anomaly detection

#### 3.4.1. Anomaly detection: the basic CUSUM

The basic CUSUM detector determines whether a parameter $\theta$ in a probability density function (PDF) has changed. It essentially chooses between two hypotheses: $H_0$:$\theta = \theta_0$ and $H_1$:$\theta = \theta_1$. Let $p_{\theta_0}$ and $p_{\theta_1}$ denote the PDF before and after the change, respectively. We do not have to be concerned with how the PDFs are determined because when we assume that the process is independent and identically distributed (iid) Gaussian, all we need to know are the mean values of the distribution and other terms cancel out. Let $y_k$ denote the $k$th sample of the data sequence (i.e. quantified sentiment sequence). The basic CUSUM decision function is

$$g_k = \max\left(g_{k-1} + \ln\frac{p_{\theta_1}(y_k)}{p_{\theta_0}(y_k)}, 0\right), \tag{2}$$

$$t_a = \min\{k: g_k \geq \bar{h}\}, \tag{3}$$

where $\bar{h}$ is threshold. Here, $t_a$ is called *stopping time*, the time when the detector identifies a change and raises an alarm. Each time when $g_k \leq 0$ or $g_k \geq \bar{h}$, CUSUM detector restarts by setting $g_k = 0$ and a new round of detection begins.

When $p_{\theta_0}$ is a Gaussian process with mean $\mu_0$, $p_{\theta_1}$ is a Gaussian process with mean $\mu_1$, and both have variance $\sigma^2$, equation (2) detects mean change and becomes

$$g_k = \max\left(g_{k-1} + \left(y_k - \mu_0 - \frac{\mu_1 - \mu_0}{2}\right), 0\right). \tag{4}$$

which can be written as

$$g_k = \max(g_{k-1} + (y_k - \mu_0 - \omega), 0) \tag{5}$$

Even if the distributions are not Gaussian, the above detector is still sensitive to mean change (Philips).

#### 3.4.2. Modified CUSUM (mCUSUM)

The basic CUSUM does not consider that the change can last for some time. It is somewhat simplistic for our purposes. Also, the above discussion on basic CUSUM is unidirectional, applicable to positive sentiment. To apply it to our problem, we enhanced it as described in Liu and Sun (2010). We extended it to both directions, particularly focusing on the negative sentiment in the other direction, the equation for which is as follows

$$g_k^- = \max(g_{k-1} + (\mu_0 - y_k - \omega), 0) \tag{6}$$

For brevity we will not repeat the MLE derivation already given in Liu and Sun (2010). The end result is captured in the algorithms below. In addition, we made a few other changes to the algorithm on top of Liu and Sun (2010) as described below.

As can be seen from the discussion on basic CUSUM above, for change detection, there are two key values: the mean before a change in sentiment and the threshold. The basic CUSUM assumes that the change occurs only after some time and uses the first few values, say the first 2, 5, or 10 values to determine the mean before change. We tried that first. However, when we were trying to optimize the performance, we discovered that using the mean for the sentiment scores of the entire corpus as $\mu_0$ gave us far better accuracy, but at the cost of not being able to apply it in realtime. The reason may be because the assumption that change occurs only after some time may not always be true. The mean or moving average can start fluctuating from the second value itself. As we see in one of the scenarios, the attack can happen before anyone even expresses their opinion. We want our solution to detect the attack in this situation as well. The limitation that the modified CUSUM cannot be applied in realtime will be addressed by the framework that we propose toward the end.

The above approach of using the mean of the scores for the entire corpus as reference will work because the corpus is expected to be huge compared to the period of attacks. Attacks can sway the overall mean to some extent, but, as we shall see, not so much as to jeopardize the detection algorithm.

As part of the experiments, we varied the threshold and the presumed mCUSUM sensitivity constant, $\omega$, which is known to be a positive value, in a nested loop to find the optimum threshold value that results in best accuracy, using the training data. Since this is a routine machine learning task, for brevity, we do not give the details of how to compute the precision, recall, and other accuracy numbers. The final

[1] http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6010/zip/imm6010.zip.

algorithms are given as Algorithms #1–#3 below. For easy reference, we call the CUSUM algorithm modified herein as mCUSUM. Algorithm #1 computes $g_k$ using the equations (5) and (6).

Algorithm 1
mCUSUM: Compute $g_k$ values.

Input: $\omega$, whose initial value is 0 and optimum value is determined by trial in Algorithm 3 and Array of sentiment scores Y, obtained as described in 3.3
Output: $G^+$ and $G^-$, Arrays of $g^+$ and $g^-$ values after applying the above equations
$\mu_0 = \frac{\sum s}{len(S)}$
/* Iterate on the sentiment scores for each of the microblogs */
**while** ($y_k \ \varepsilon \ Y$) **do**
  $g_k^- = \max(g_{k-1} + (\mu_0 - y_k - \omega), 0)$
  $g_k = \max(g_{k-1} + (y_k - \mu_0 - \omega), 0)$
**end while**
$G^+ = [g_k^+]$
$G^- = [g_k^-]$

Algorithm #2 finds the fake tweets using the starting and stopping time equations for the modified mCUSUM.

Algorithm 2
mCUSUM: Find Fake Microblogs.

Input: Threshold, $\tau$ and Array of sentiment scores Y.
Output: IFA, the confusion matrix (or array) indicating fake microblogs; IFA[i] = 1 if the microblog is fake.
**while** ($y_k \ \varepsilon \ Y$) **do**
  /* The goal is to find indices start and stop repeatedly until we iterate on all microblogs. These two indices indicate the interval of attack */
  start = stop = 0
  **while** (i > 0 and i < *len*(*Y*) and $g_k^- > \tau$) **do**
    **if** (not start) **then**
      at1 = i
      **for** j in reversed(xrange(i)) **do**
        **while** (j > 0 and $g_j^- >= g_{j-1}^-$) **do**
          j - = 1
        **end while**
        start = j + 1
        break
      **end for**
    **end if**
    i + = 1
    **if** (start) **then**
      at2 = i
      stop = m where at1 < = m < =at2 and
  $g_m^- = \max(G^-[at1:at2])$
      *IFA*[start:stop] = [1]
      *return*(*IFA*)
    **end if**
  **end while**
**end while**

Algorithm #3 computes the optimum threshold and $\omega$ values and feeds them into the above two algorithms to find the fake tweets. The probability $p$ in our problem statement is the value returned by *findAUC*() in the algorithm, using the final values of $\tau$ and $\omega$.

Fig. 2 shows the sentiment scores for the tweets plotted in red in the above half and the computed values of $g^-$ plotted in green in the lower half. The sentiment scores are also plotted as a line in blue in the lower half for contrast. The optimum threshold determined for this scenario is quite close to 0, hence the line for the threshold is not clearly seen. As can be seen, the sentiment scores vary between −5 on the negative side to 3 on the positive side, with most of scores in the negative. The green line in the bottom plot is what is used to determine which tweets are fake, as we will see. An attack is not readily evident by looking at the plots either at the top or bottom. The relationship between the green line at the bottom and the red points at the top that is given in equation (6) is also not easily recognized from the plots. But as we shall see, the plots are from one of the scenarios of attack and the mCUSUM algorithm is quite successful in detecting the attack.

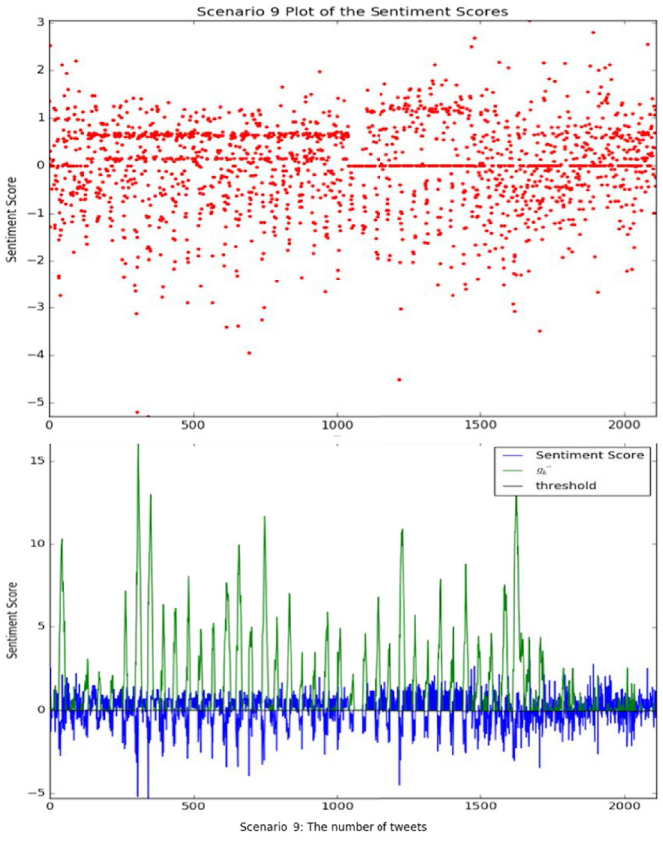


**Fig. 2.** Sentiment Score and the corresponding mCUSUM Plot for a Scenario; we can see how $g_k^-$ varies with the changes in sentiment score. Scenario 9, described later in the paper, is chosen randomly - it can be any other scenario for the purpose of illustration.

Algorithm 3
Finding the Optimum Threshold, $\tau$, mCUSUM constant, $\omega$, and the Fake Tweets Array, IFA.

Input: Array of sentiment scores Y
Output: IFA, Array of fake microblogs
// Iterate on the threshold
**for** ($\tau_{temp}$ = 0; $\tau_{temp}$ < = 25; $\tau_{temp}$ + = 0.25) **do**
  // Iterate on the mCUSUM constant, $\omega$
  **for** ($\omega_{temp}$ = 0; $\omega_{temp}$ < =1; $\omega_{temp}$ + = 0.05) **do**
    /* Run the first part of the mCUSUM algorithm to fill the 'g' values in array, $G_{temp}^-$, using the array of given sentiment scores *Y* */
    $G_{temp}^-[\tau_{temp}, \omega_{temp}]$ = *cusum* ($\omega_{temp}$,Y)
    /* Run the second part of the mCUSUM algorithm on $G^-$ to get the array of fake tweets, IFA */

(*continued on next page*)

Algorithm 3 (*continued*)

$IFA_{temp}[\tau_{temp}, \omega_{temp}] = findFakes\ (\tau_{temp}, G_k^-)$
/* Find the Area Under the ROC curve for these values of threshold and $\omega$ */
$A_{temp}[\tau_{temp}, \omega_{temp}] = findAUC\ (IFA_{temp})$
**end for**
**end for**
/* Get the values of the threshold and $\omega$ which maximize the accuracy of the prediction measured by the Area Under the ROC curve, based on the already labeled tweet data */
$\tau, \omega = getKeys\ (\max\ (A_{temp}))$
/* Using the above values run the mCUSUM algorithm one final time to find the fake microblogs */
$G^- = cusum\ (\omega, Y)$
$IFA = findFakes\ (\tau, G^-)$

### 3.4.3. Statistical anomaly detection: discrete Kalman Filter

An excellent discussion on Kalman Filter is given in Welch and Bishop (2001) and quite a few other publications as well, so we will not go into the details of the theory. The general system of Kalman Filter equations can be transformed into the following single variable case, given by equations in Welch and Bishop (2001).

$$\hat{x}_k^- = \hat{x}_{k-1} \tag{7}$$

$$P_k^- = P_{k-1} + Q \tag{8}$$

$$K_k = \frac{P_k^-}{P_k^- + R} \tag{9}$$

$$\hat{x_k} = \hat{x_k}^- + K_k(z_k - \hat{x_k}^-) \tag{10}$$

$$P_k = (1 - K_k)P_k^- \tag{11}$$

In the above equations, (7) and (8) comprise the time update and the remaining 3 equations comprise the measurement update. $z_k$ are the sentiment scores of individual microblogs. $x_k$ are analogous to the $g_k$ in mCUSUM and have to be computed. For simplicity, we assume Q and R to be constant, just like we did for $\omega$ in mCUSUM. Just like we computed the optimum values for $\omega$, the threshold $\tau$ by iterating through a number of possible values, we do the same in case of Kalman Filter, by iterating through a number of possible values for the offset $\tau$ and constants Q, and R, using the training data.

Initial seed values for P, K and $\hat{x}_k$ are assumed to be 0. Algorithms #4 and #5 below directly apply the Discrete Kalman Filter equations to detect fake tweets.

A plot of the sentiment scores of the tweets after applying the Kalman Filter algorithm is shown in Fig. 3. The dots in green, above the red line for $\tau$ are presumed to be genuine and the ones marked with a '+' in maroon below the red $\tau$ line are detected to be part of an attack. The scenario is that of injecting twelve fake tweets after a random number of real tweets not exceeding forty. As can be seen from the figure, Kalman Filter correctly identified most of the twelve maroon '+' for every forty or less green dots as the scenario entails. There are of course false positives. These and other results are discussed in detail later. In this scenario, the optimum $\tau$ was determined to be $-0.2$ and is shown by the red line. The line for the expected sentiment is shown in blue, above the red line.

Algorithm 4
*xhatCompute*: Determine Expected Sentiments using Kalman Filter Equations.

[1]
// Direct application of the Kalman Filter Equations
**while** ($z_k \varepsilon Z$) **do**

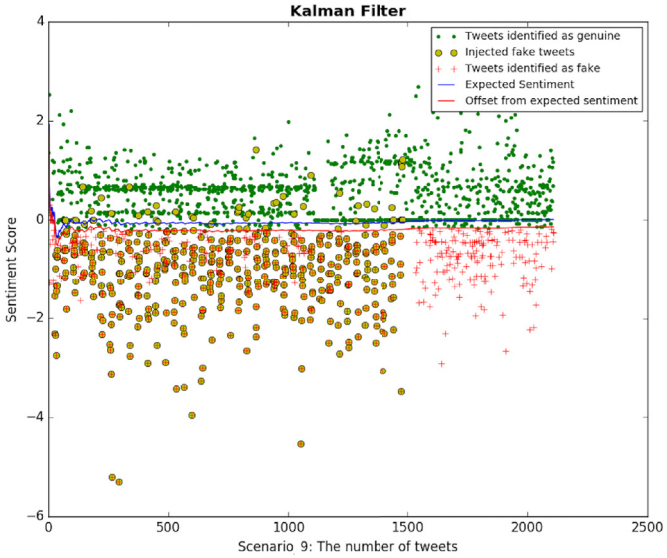


**Fig. 3.** Kalman Filter results for the same scenario as for Fig. 2; '+' in red are identified as fake tweets; red line is the offset from the expected sentiment drawn in blue; green dots are identified as genuine. Scenario 9, described later in the paper, is chosen randomly - it can be any other scenario for the purpose of illustration.

Algorithm 4 (*continued*)

// Time update
$\hat{x_k^-} = \hat{x_{k-1}}$
$P_k^- = P_{k-1} + Q$
// Measurement update
$K_k = \frac{P_k^-}{P_k^- + R}$
$\hat{x_k} = \hat{x_k}^- + K_k(z_k - \hat{x_k}^-)$
$P_k = (1 - K_k)P_k^-$
**end while**

Algorithm 5
Determining Optimum values of Q, R, and offset $\tau$.

// Iterate on the possible values for $\tau$, $R$, and $Q$
**for** ($\tau_{temp}$ = -0.5; $\tau_{temp} < = 0$; $\tau_{temp} + = 0.05$) **do**
**for** ($R_{temp} = 0$, $R_{temp} < = 0.02$; $R_{temp} + = 0.001$) **do**
**for** ($Q_{temp} = 0$, $Q_{temp} < = 1e-4$; $Q_{temp} + = 1e-5$) **do**
$\hat{x_k} = xhatCompute(R_{temp}, Q_{temp}, Z)$
**while** ($z_k \varepsilon Z$) **do**
**if** ($z_k$ - $\hat{x_k} < \tau_{temp}$) **then**
$\hat{y_k} = 1$
**end if**
**end while**
// Just like for mCUSUM compute the accuracy of results for each case
$AUC_{temp} = findAUC(\hat{Y_{temp}}, Y)$
**end for**
**end for**
**end for**
$\tau, R, Q = getKeys((\max(AUC_{temp})))$
$\hat{x} = xhatCompute(R, Q, Z)$
**while** ($z_k \varepsilon Z$) **do**
**if** ($z_k$ - $\hat{x_k} < \tau$) **then**

(*continued on next page*)

Algorithm 5 (*continued*)

$\hat{y_k} = 1$
**end if**
**end while**

### 3.5. Classification as real or fake post

The parameters computed during the statistical anomaly detection module are used to classify the microblogs using the algorithms given in this section. The algorithms are specific to techniques used in the previous model, so are included in the above section for ease of understanding.

## 4. Experiments

Realtime experiments using Twitter APIs, like Andres Sepulveda did, could actually land one in trouble with the law. We therefore implemented the offline framework that integrates mCUSUM and Discrete Kalman Filter.

### 4.1. Dataset

To proceed with the experiments, we need an offline corpus *M* of microblogs covering a major event *T*, preferably a Presidential election, in view of the impact of the event and possible chances of an influence attack *IFA*. The advantage of an offline corpus is the ease with which an attack can be faked. A simulated attack will then be a matter of text modification than using APIs to actually post the microblogs to sway the influence. We tried using Twitter streaming APIs to collect live posts over a few weeks, but the collection was not sufficiently big enough on any single topic, particularly the US Presidential election that we wanted to focus on. We therefore searched online and found a data set of around 10 million tweets used for a paper in the CIKM conference.[2] We analyzed the 10 million tweets and identified the following four socially sensitive topics during the period based on the relative frequencies of the hashtags:

1. Iran Elections[3]
2. Tea Party[4]
3. Haiti Earthquake[5]
4. Yankees[6]

We did an automated sentiment analysis of all the four categories of the tweets as described in 3.3. The sentiment scores for the tweets were tabulated and plotted for visual analysis. Of the above four, the tweets related to the 2009 Iran Presidential elections, numbering 1650 seemed to be ideally suited for our project, based on our goal and the sentiment fluctuations in the corpus itself. Now we have *M* and *T*.

### 4.2. Injecting influence

Next, we need to inject negative influence *IFA* into these microblogs to simulate Andres Sepulveda's “Social Media Predator” program's attack *A*, but offline. Malicious accounts try to make their tweets diverse to avoid being detected as robots. To generate diverse microblogs with negative sentiment, we tried to build on top of a few random sentence generators based on Hidden Markov Models using a set of words and articles with negative sentiment that relate to the Iran elections. We tried NLTK (Bird et al., 2009) based approaches as well, but the semantic quality of the sentences generated using all these approaches turned out to be poor.

As an alternative approach, we manually collected a series of online articles containing negative sentiment on the topic. We wrote a python program to use the Affective Lexicon (AFINN) (Nielsen, 2011) to analyze the sentiment of the sentences in the article. Sentences from the articles with substantially negative sentiment were then selected by the program and fed to another program that uses a simple markov chain approach, which seemed to be popular with twitter bots. The latter program generated random and reasonably sensible sentences, which were compacted into microblogs of 140 characters or less. We used these negative microblogs, numbering around 460, for injection into the real world corpus to simulate various scenarios.

### 4.3. Scenarios

The experiments are organized to model various typical real world scenarios as described in this section. These scenarios typically arise from deliberate, injected attacks and not usually a result of, say, a hot mic or a genuine leak. For instance, below is a function call to a typical twitter bot that seems to be popular online:

```
tweetbot.twittertweetingstart(days=0,
  hours=19, minutes=30, keywords=None,
  prefix=None, suffix='#PyGaze')
```

The above scenario is covered by scenario 5 below. Other scenarios are similarly designed to simulate how attacks can happen in reality. It may however be true sometimes that some of the scenarios may have been caused by genuine happenings and not deliberately injected. Such occurrences will have to be correlated to news media or use some other techniques to rule out an injected attacked. This paper does not deal with these corner cases. It can be reasonably assumed that the public opinions on a specific topic do not change dramatically within a short time window, without a deliberate attempt to inject negative influence.

The scenarios are also designed to take into consideration that not all sentiment changes can be attributed to a malicious attack. There is a chance that the performance of the algorithms could be impacted by false positives. The scenarios are specifically designed also for negative testing the algorithms and analyze the false positives rate.

We organize various scenarios in four different categories according to how fake tweets are injected. The detailed discussions are explained below.

#### 4.3.1. Attack timing

Scenario #1. Fake tweets all at once, right at the beginning, followed by several real tweets. This covers the case where the attacker wants to preempt the influence on a topic by taking the lead.

Scenario #2. All fake tweets at once after a number of real tweets. The results for both mCUSUM and Kalman Filter are same as in scenario 1. This implies that timing does not matter from the algorithm perspective, but may matter in reality because of the influence on other tweets. This realtime influence fluctuation could not be tested because our data set did not include a live attack or live users.

The plots for mCUSUM and Kalman Filter for this category are shown in Figs. 4 and 5 respectively. Both the algorithms detect the attack with high accuracy. The mCUSUM algorithm detected almost all of the fake tweets, achieving a 99.9% accuracy. The plot in Fig. 4 clearly shows a rising $g_k^-$ for around 460 tweets - the period of the attack, irrespective of when the attack starts. In the figure at the top for scenario #1, the attack started right away, causing $g_k^-$ to rise until the attack ended. In the second scenario at the bottom, the attack started after about 400 tweets, when $g_k^-$ also started rising until the attack ended.

The Kalman Filter algorithm on the other hand achieved 83.2%

[2] https://archive.org/download/twitter_cikm_2010/twitter_cikm_2010.zip.
[3] https://fas.org/sgp/crs/mideast/R40653.pdf.
[4] https://en.wikipedia.org/wiki/Tea_Party_movement.
[5] http://www.cnn.com/2010/TECH/01/14/twitter.hoax.haiti/.
[6] https://en.wikipedia.org/wiki/Yankee_Stadium.

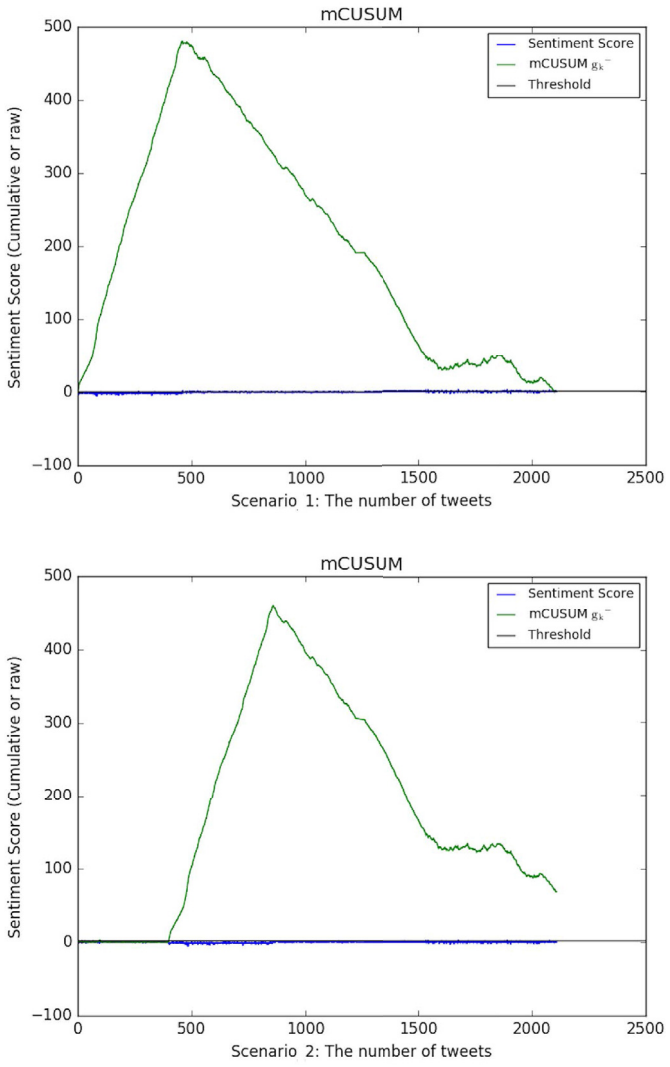


**Fig. 4.** mCUSUM plots for the scenarios examining the effect of attack timing; the detection results for these scenarios are almost perfect. The period of the steep rise of $g_k^-$ to its maximum, is the period of the attack.

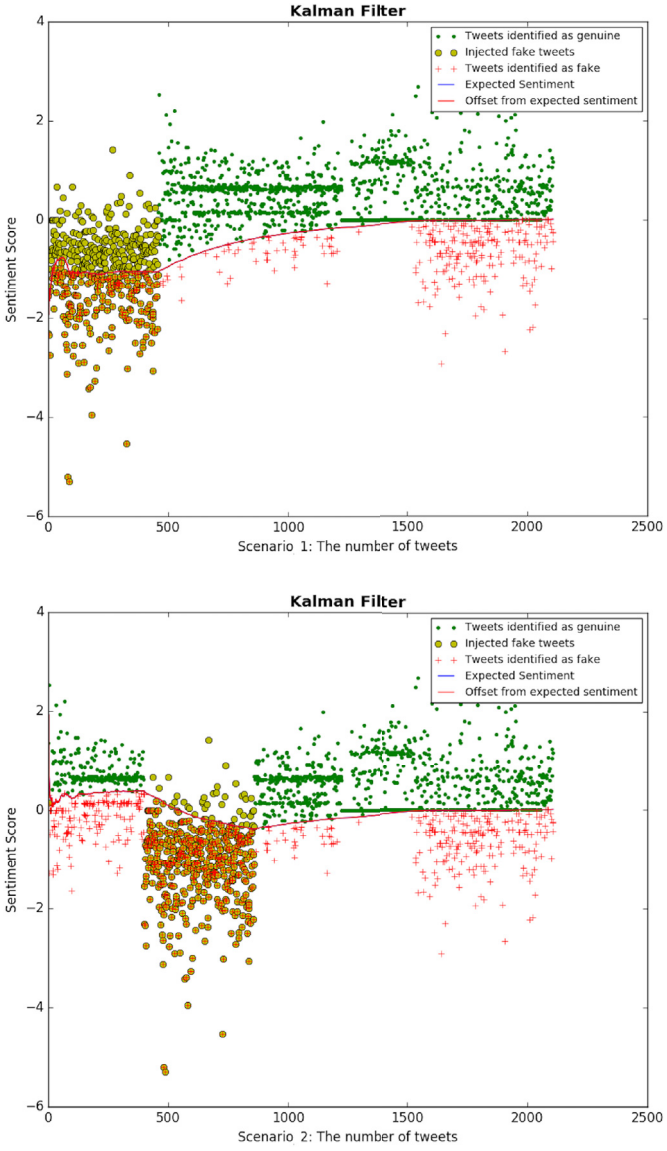


**Fig. 5.** Kalman Filter plots for the scenarios examining the effect of attack timing.

accuracy. It identifies quite a few tweets as fake, incorrectly. This is because the Kalman Filter is based on expected sentiment that recursively depends on the previous state. When the attack all happens at once, the expected sentiment is negative during the attack, but gets back into the positive region soon after. Most negative tweets, even if genuine, therefore get identified as fake if they are not in the tolerance offset from the expected sentiment score. This can be observed from Fig. 5.

### 4.3.2. Viciousness

Scenario #3. The fake tweets alternating with real tweets right from the beginning. The attacker monitors every genuine tweet and responds to it negatively, a scenario quite common in the reality.

Scenario #4. Fake tweets alternating with real tweets after certain number, say 400 of real tweets on the topic. The attacker watches for some time, gauges the sentiment and decides to start swaying the influence by posting a negative tweet for every genuine tweet.

The plots for mCUSUM and Kalman Filter for the viciousness category are shown in Figs. 6–8, 9 respectively. In the Scenario 3 plot for mCUSUM in Fig. 6, we can see that $g_k^-$ mostly rises as long as the attack lasts and then changes direction immediately. There are many small changes in direction along the green line, indicating the alternate sentiment shifts designed in the scenario. Since the attack comprises of

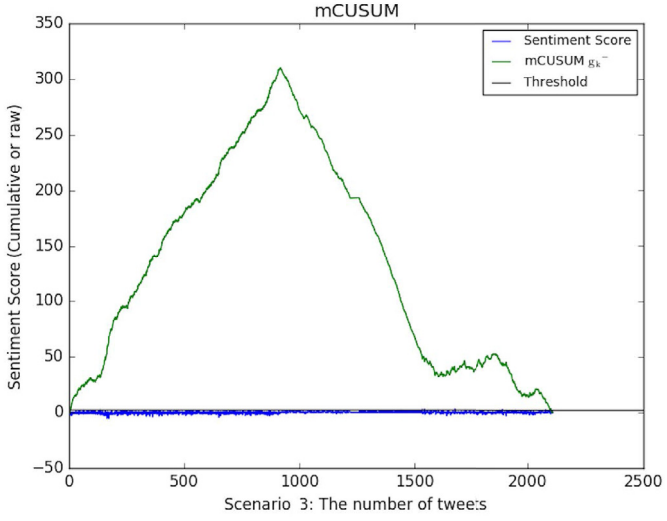


**Fig. 6.** mCUSUM plots for the Viciousness Scenario 3.

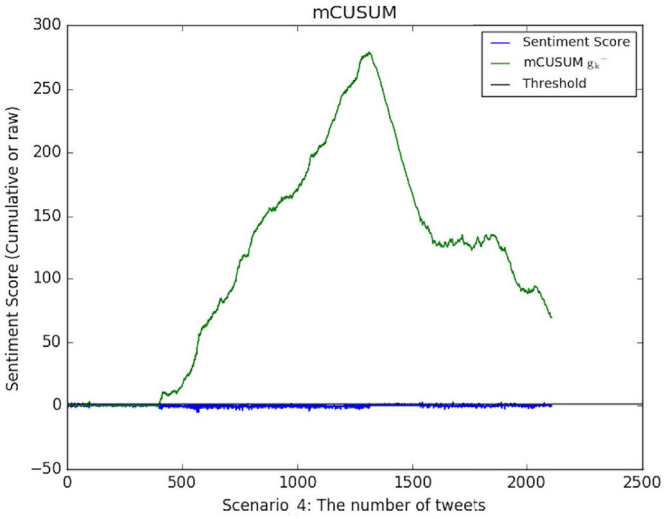


**Fig. 7.** mCUSUM plots for the Viciousness Scenario 4.

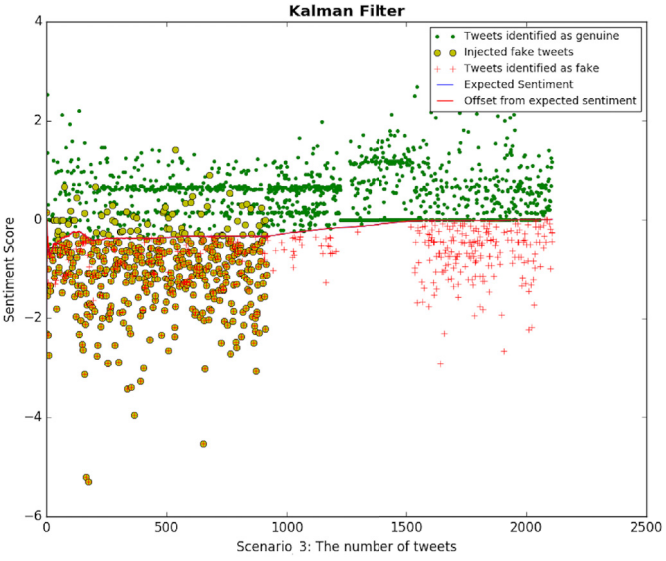


**Fig. 8.** Kalman Filter plots for the Viciousness Scenario 3. The expected sentiment is hidden behind the red offset line because the offset in this case is 0. Because of the preponderance of positive tweets, the expected sentiment is mostly positive, causing most negative tweets to be identified as fake, even if genuinely posted.

around 460 fake tweets, the stopping time indicated by the change of direction occurs at around 920th tweet, given that the fake tweets alternate with the genuine tweets. It is apparent that the genuine tweets did not contribute much to the sentiment shift because the overall impact of the attack has $g_k^-$ rising to its peak. We also notice a few cusps toward the end, although there was no attack at that time. These direction shifts can be attributed to genuine opinions.

The mCUSUM plot for the fourth scenario in Fig. 7 is quite similar to the third, just that the attack starts after around 400 tweets. In this case too, from the shape of $g_k^-$, it can be seen that the algorithm has correctly identified most of the fake tweets. It must be noted that the threshold is not clearly seen in the plots because it is too close to the X-axis.

The Kalman Filter plot in Figs. 8 and 9, for the scenarios is straight forward. Each point on the plot is a raw sentiment score for one of the tweets. The yellow circles are the sentiment scores of injected fake tweets. They are mostly negative, so are predominantly in the lower half of the Cartesian plane. The green dots are the sentiment scores of

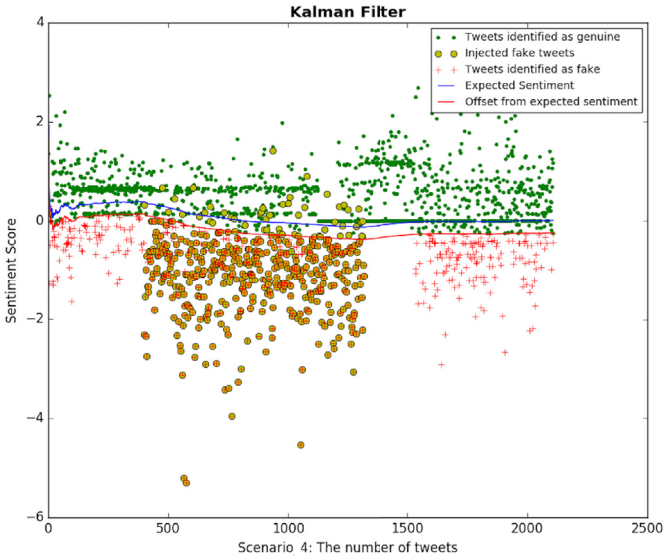


**Fig. 9.** Kalman Filter plots for the Viciousness Scenario 4.

the tweets that the Kalman Filter has identified as genuine and the red "+" points are those of the fake tweets. A red "+" on a yellow circle indicate the fake tweets that the algorithm has correctly identified. We plot the sentiment score that we expect based on the previous tweets as a blue line and the offset from it as a tolerance limit plotted as a red line. The points below the red line are identified by Kalman Filter as fake tweets.

As can be seen from Figs. 8 and 9, the Kalman Filter identifies most of the fake tweets correctly, but is impacted by a number of false positives as well. The reason is the Kalman Filter's recursive dependence on the previous state to compute the expected sentiment. Because of the preponderance of genuine, positive sentiment tweets, the expected sentiment is mostly positive, causing most negative sentiment tweets to be identified as fake, even if they are genuinely posted.

In the plot for Scenario 3, the injected fake tweets can mostly be seen to the bottom left, alternating with the green tweets, which are mostly on the top. The Kalman filter identified most of the fake tweets, but there are quite a few false positives as well to the right. The algorithm correctly identified most of the genuine tweets. The blue line representing the expected sentiment merged with the offset because the tolerance limit for this scenario has been determined to be 0. It is clear from the figure that the fake tweets have been injected only till the 920th tweet, since there were around 460 fake tweets alternating with the genuine tweets. The Kalman Filter plot for Scenario 4 can be similarly interpreted. Both the blue line for the expected sentiment and the offset for tolerance in red can be seen in this plot. The attack starts with a delay, but Kalman Filter identified a few false positives right at the beginning. It can also be seen that the algorithm correctly identifies most of the fake tweets and the genuine ones.

#### 4.3.3. *Weak attack power*

Scenario #5. One fake tweet after a random number of genuine tweets not exceeding a fixed number, say 40. That means only around 41 fake tweets for 1650 real tweets. This scenario is often achieved by employing a bot. This scenario tests the case of weak influence. Both mCUSUM and Kalman Filters are effective. Implies that even weak influences can be detected.

Scenario #6. Three fake tweets after a random number of real tweets not exceeding a fixed number, say 40. This implies only 123 fake tweets for our corpus of 1650 real tweets. This further tests the case of weak influence.

Scenario #7. Up to six random number of fake tweets after a random number of real tweets not exceeding, say 40. This implies only

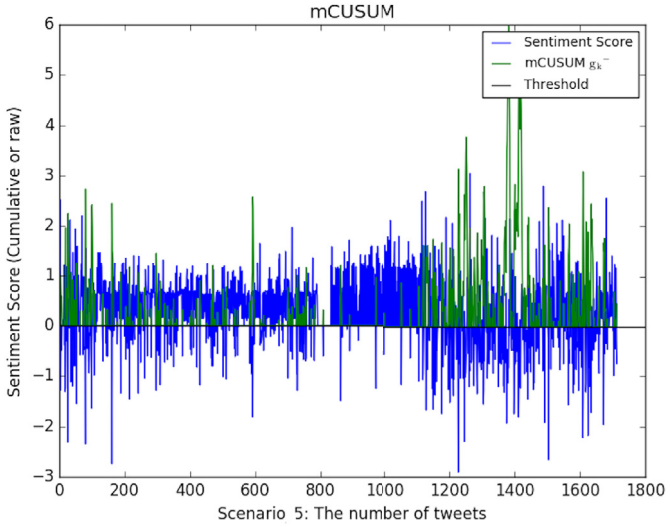


**Fig. 10.** mCUSUM plots for the scenario 5 covering weak attack power.

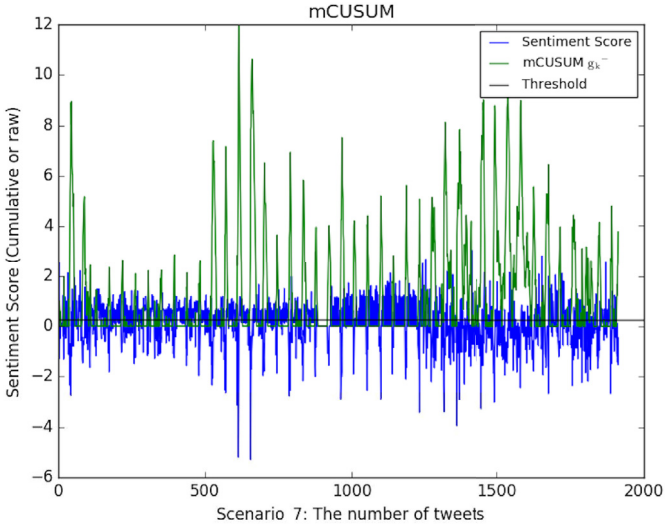


**Fig. 12.** mCUSUM plots for the scenario 7 covering weak attack power.

246 fake tweets for 1650 real tweets. This also is a test for the case of weak, but increased influence.

The plots for mCUSUM and Kalman Filter for category examining the weak influence are shown in Figs. 10–13, 14, 15 respectively. Key things to note from the mCUSUM plots are that the injected fake tweets are far and few, hence the cumulative negative sentiment represented by the green line is far less compared to the previous scenarios. The green line fluctuates and changes its direction at most places the fake tweets are injected, implying that the mCUSUM identified the injected fake tweets correctly, most of the time. Another point to note is that the rise of the green line corresponds to the spikes of the sentiment score in the opposite direction, giving the impression of a reflection. This is because the blue line spikes to the bottom mostly indicate fake tweets with negative sentiment. The green line captures the cumulative sum of the sentiment score and changes direction when negative sentiment attack is done with.

The Kalman Filter plots clearly indicate the injected fake tweets with a yellow circle, as before. The blue line representing the expected sentiment and the offset from it represented by the red line are distinctly visible in the weak influence scenarios, giving more benefit of doubt to the tweets, since the influence is weak, which is quite intuitive. Though most of the injected tweets are detected correctly, as indicated

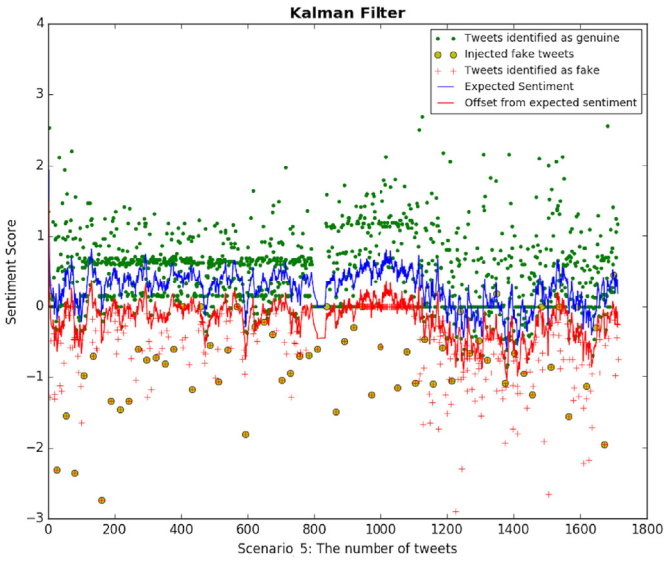


**Fig. 13.** Kalman Filter plots for the scenario 5 covering weak attack power.

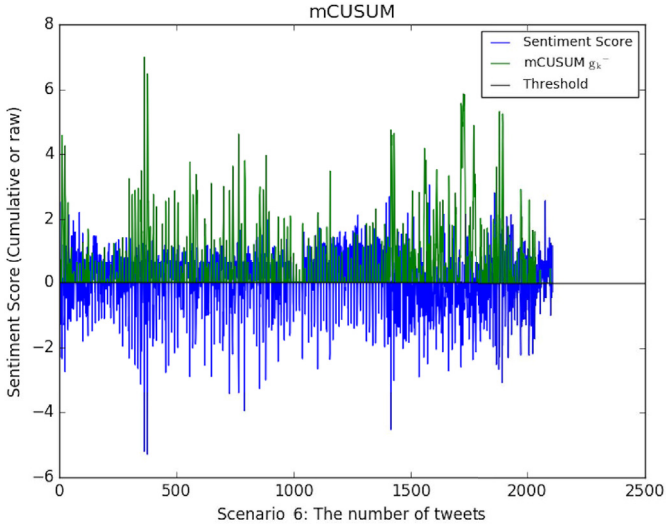


**Fig. 11.** mCUSUM plots for the scenario 6 covering weak attack power.

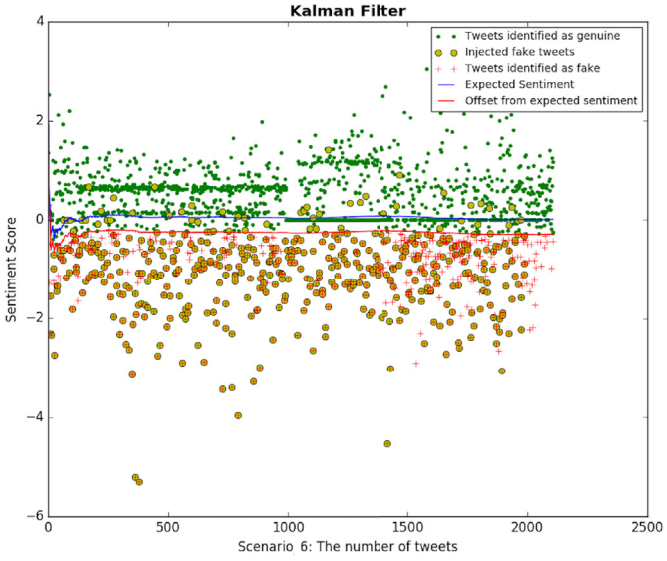


**Fig. 14.** Kalman Filter plots for the scenario 6 covering weak attack power.

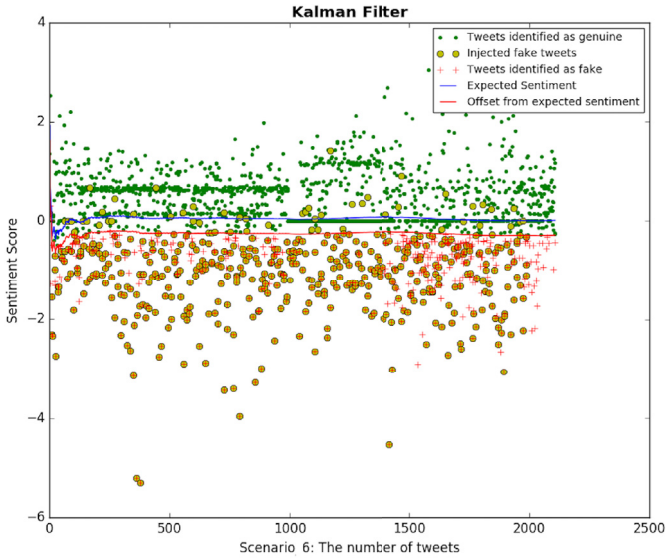


**Fig. 15.** Kalman Filter plots for the scenario 7 covering weak attack power.

by a red "+" inside the yellow circle, there are also a number of false positives. It can also be seen that most of the genuine tweets have also been identified correctly. Performance metrics are discussed later in the section on "Algorithm Performance Analysis". As can be seen from various figures, the algorithms are impacted by false positives differently, based on the scenario and the attack pattern. The reasons are explained later, towards the end of the "Algorithm Performance Analysis" section.

#### 4.3.4. Random attacks

Scenario #8. Nine fake tweets after a random number of genuine tweets not exceeding 40. Implies 369 fake tweets for 1650 real tweets. This and the previous scenarios tell us how the algorithms fare with gradually increasing influence.

Scenario #9. Twelve fake tweets after a random number of real tweets not exceeding 40. Tests periodic and sporadic attacks, typically using a microblog bot.

Scenario #10. Six fake tweets after a random number of real tweets not exceeding 20. Tests the impact of increasing frequency of attack.

Scenario #11. One fake tweet after a random number of real tweets not exceeding 4. This tests the impact of sporadic influence attacks.

Figs. 16–19 show the plots for the mCUSUM runs for this category of

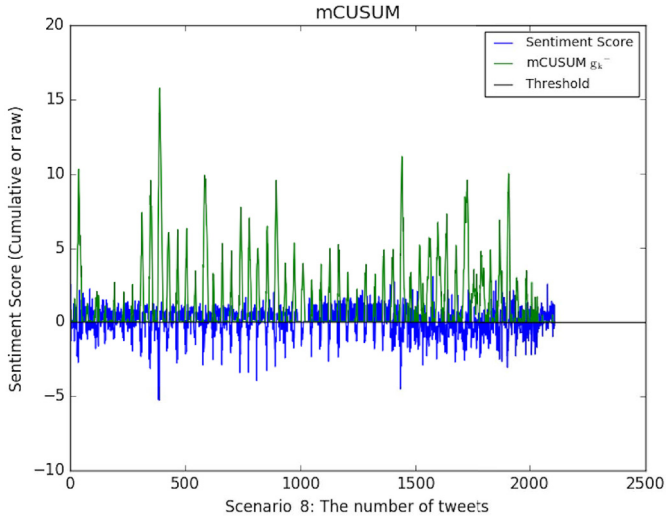


**Fig. 16.** mCUSUM plot for scenario 8 examining the effect of random attacks.

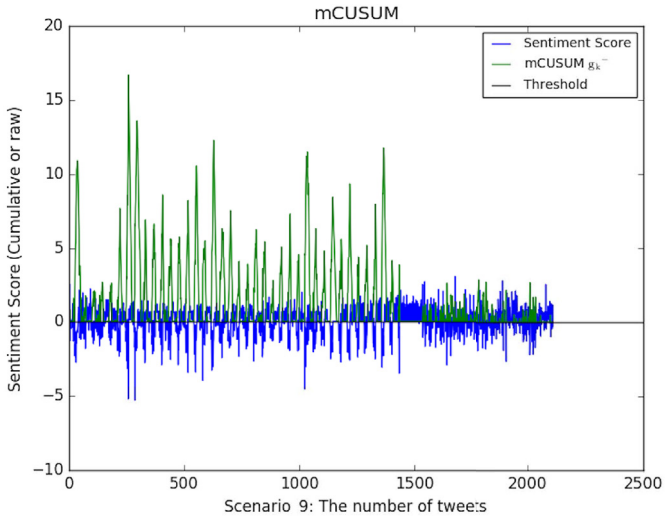


**Fig. 17.** mCUSUM plots for scenario 9 examining the effect of random attacks.

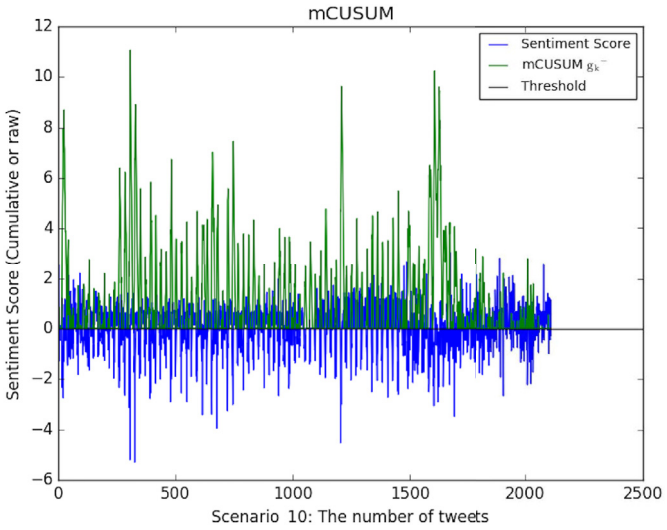


**Fig. 18.** mCUSUM plots for scenario 10 examining the effect of random attacks.

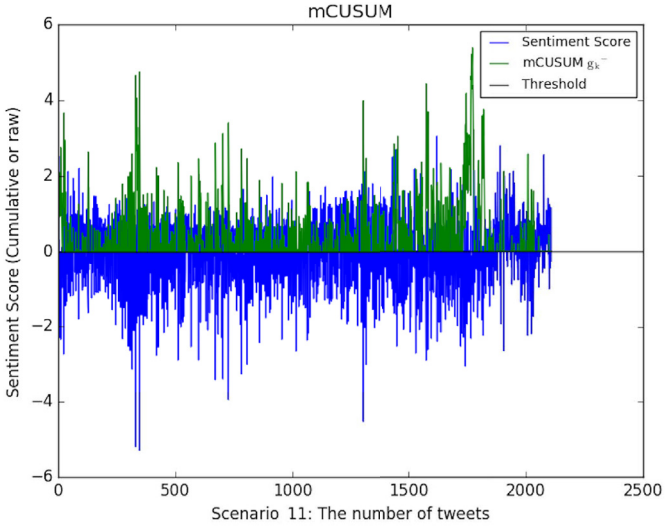


**Fig. 19.** mCUSUM plots for scenario 11 examining the effect of random attacks.

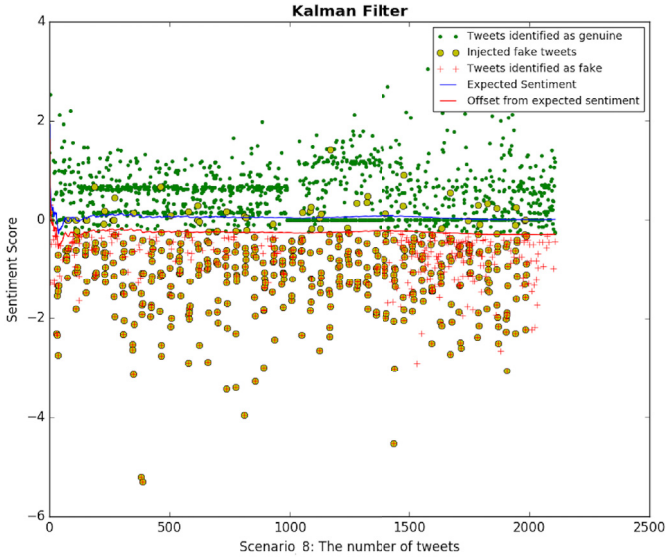


**Fig. 20.** Kalman Filter plots for scenario 8 examining the effect of random attacks.

scenarios. The sentiment fluctuations in each scenario are visible from the plots. The plots resemble the ones for weak influence. However, since a bunch of fake tweets are posted together in this case, the cumulative effect is much more profound as the heights of the green lines demonstrate. The more closer the fake tweets are posted, the higher is the cumulative impact of the period. The observed behavior is quite along the lines of the intuitive reasoning.

The plots for the Kalman Filter runs for the same scenarios are in Figs. 20–23. Each figure illustrates the performance of the algorithms for the respective scenario. The scenarios are all different attack patterns, resulting in different behavior and metrics. Once again, it can be seen that most of the identification happens correctly, although there are a few false positives represented by the red "+" points not enclosed in a yellow circle. It can also be seen that the expected sentiment and the offset from it are clearly seen, indicating some tolerance to sentiment fluctuations.

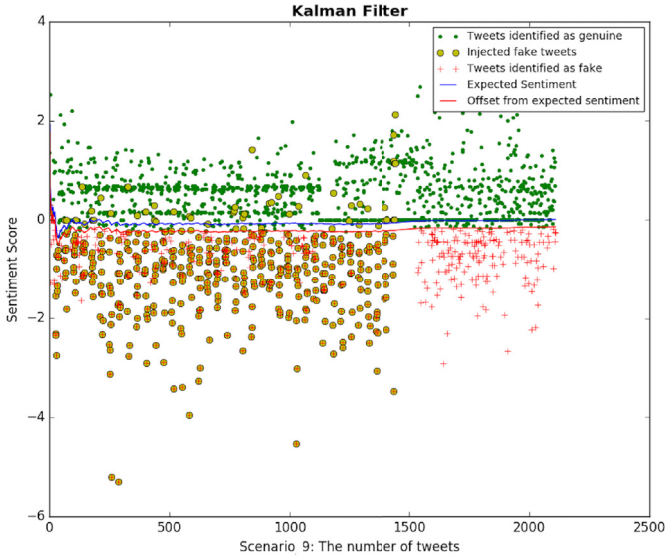


**Fig. 21.** Kalman Filter plots for scenario 9 examining the effect of random attacks.

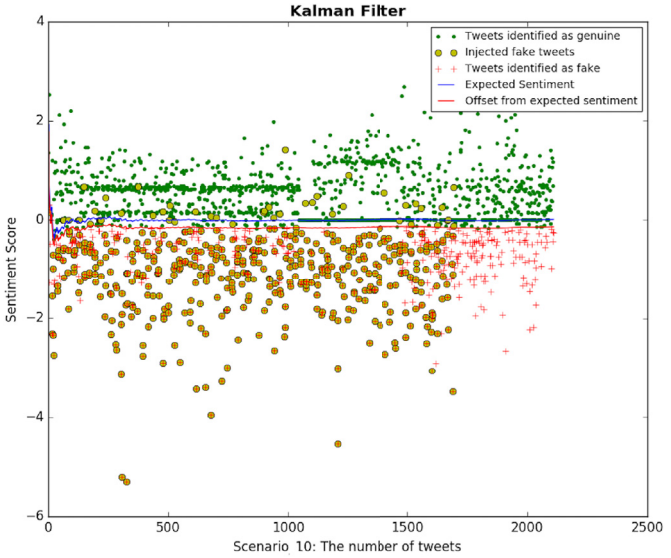


**Fig. 22.** Kalman Filter plots for scenario 10 examining the effect of random attacks.

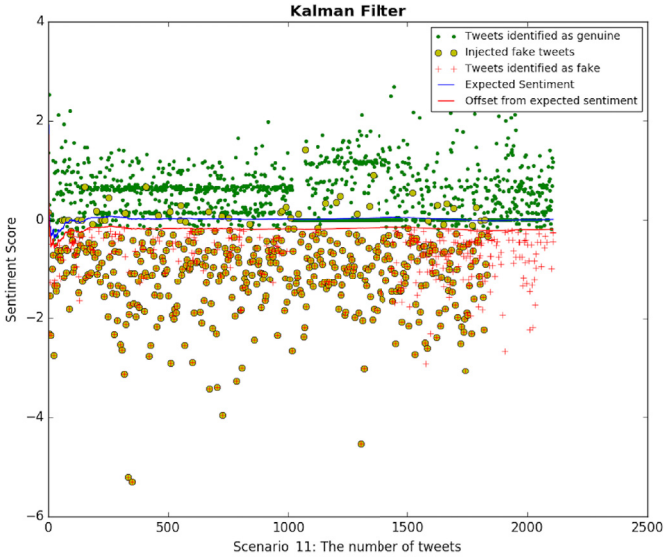


**Fig. 23.** Kalman Filter plots for scenario 11 examining the effect of random attacks.

**Table 1**
Evaluation Metrics for one Scenario examining the effect of attack timing. These metrics were computed for all scenarios. For brevity, only Scenario #2 has been randomly chosen to illustrate the metrics. Any other scenario would have served the purpose.

| Metric / Detail | mCUSUM | Kalman Filter |
|---|---|---|
| Threshold / Offset | 2 | −6.9388 |
| K | 0.05 | R = 0.0120, Q = 0 |
| Fake Tweets Injected | 459 | 459 |
| Total Number of Tweets | 2109 | 2109 |
| AUC | 0.9990 | 0.8326 |
| True Positives | 459 | 420 |
| False Positives | 3 | 412 |
| F1 score | 0.9967 | 0.6506 |
| Precision | 0.9935 | 0.5048 |
| Recall | 1 | 0.9150 |
| Avg Precision | 0.9967 | 0.7191 |

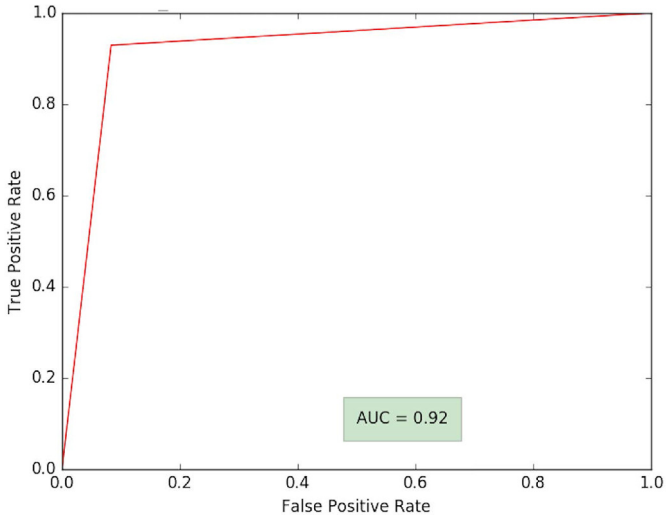


**Fig. 24.** Performance of mCUSUM applied to Scenario 9 plotted in ROC space.

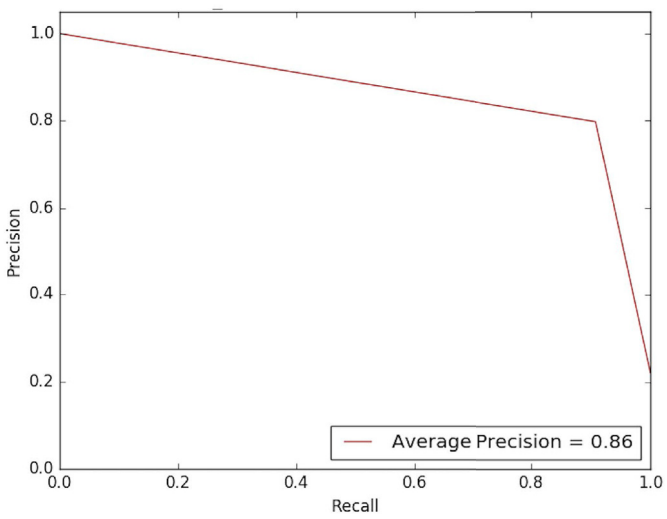


**Fig. 25.** Performance of mCUSUM for the same Scenario 9 plotted in the Precision / Recall space.

### 4.4. Algorithm Performance Analysis

A table of all the evaluation metrics we adopted to measure the performance of the algorithms and the corresponding statistical parameters is given in Table 1. The table is for one scenario. Results for other scenarios are collectively presented and analyzed later. This one scenario is for illustrating the various metrics in detail. In this particular scenario, #2, chosen for no specific reason, mCUSUM identified almost all 459 fake tweets resulting in a recall of 1 and an $F_1$ score of 0.997. Kalman Filter identified 420 out of the 459 fake tweets, but is impacted by a number of false positives, which drops its performance.

The third section of Table 1 gives the Precision, Recall, and the metrics based on these, namely, $F_1$ score and Average Precision. As can be seen, though Recall is high for Kalman Filter, the $F_1$ score or F-measure, which is the harmonic mean of Precision and recall, is low because of the low precision.

For detection problems, the standard evaluation criteria include Receiver Operating Characteristic (ROC) curve, which describes the relationship between detection rate and false alarm rate, as well as precision and recall. We chose ROC curve and precision / recall to do the comparison. Fig. 24 shows the performance of mCUSUM in ROC space for scenario #9. The x-axis measures the false positive rate, given as (1 − *specificity*), indicating how liberal or less specific the algorithm is in classifying the tweets. Specificity is the true negative rate and is a measure for the proportion of fake tweets that are correctly identified as such.

Y-axis plots the true positive rate, which is the same as recall, also known as sensitivity. The closer the curve is to the point (0,1), the greater the Area Under the curve (AUC) and the better the performance of the algorithm, because (0,1) is the point when there are no false positives and all true positives. The variable that the curve plots is the ability of the algorithm to classify the tweets correctly. The python library we used plots the performance of the algorithm in the ROC space rather than plot the points for varying thresholds. The resulting Fig. 24 shows the parameters involved in computing the AUC. The algorithm's performance, plotted on a Precision Recall curve for the same scenario is shown in Fig. 25.

Recall rate is satisfactory for most scenarios, but precision is not as good in some cases because of false positives. mCUSUM accumulates the difference between the expected value and the observed value. The expected value is computed based on the entire corpus and not just the previous tweets. When the fake tweets all come at once, the accumulated differences are substantial and can be clearly differentiated from the genuine tweets.

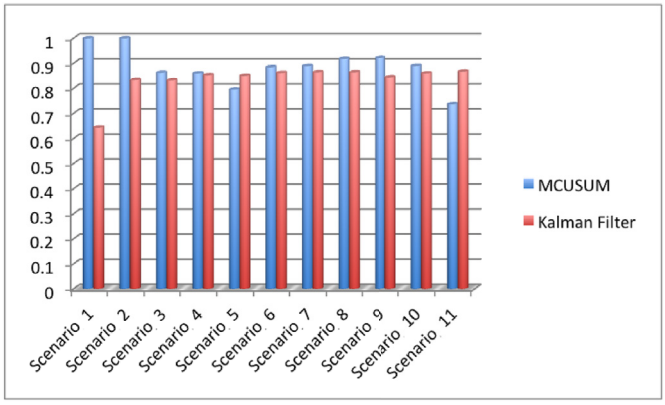


**Fig. 26.** Values of the Area Under the ROC Curve are plotted on the Y-axis for each scenario.

The following two subsections further analyze the performance of the algorithms with respect to AUC and false positive rates.

#### 4.4.1. AUC analysis

Fig. 26 shows the comparison of the Area Under the ROC Curve (AUC) for the algorithms for various scenarios. AUC is a good measure for the performance of algorithms. Higher AUC implies that the algorithm has done a good job at distinguishing between injected fake tweets and the genuine ones. As can be seen, both the algorithms performed quite well on the data in all the scenarios and the results are meaningful.

As expected, from Fig. 26, mCUSUM worked accurately (AUC = 0.999) for the scenarios where the fake tweets all come at once, irrespective of the timing. In case of Kalman Filter, the expected sentiment recursively depends on the previous state. In the first scenario, the attack starts right at the beginning. There are no genuine tweets before the attack. Kalman Filter, therefore does not know what scores can be classified as genuine, although the attack happens in the beginning. It is for this reason and reasons explained in previous sections that Kalman Filter does not perform as well as mCUSUM for the first few scenarios.

For scenarios 5 to 11, instead of injecting fake tweets in regular intervals, we used a random interval to make it closer to the real life scenario and performance of the tools is still good. Kalman Filter outperforms mCUSUM in two of these scenarios, #5 and #11, where the

fake tweets are interleaved with the genuine ones closely, because mCUSUM is not sensitive enough when weak attacks occur periodically. Kalman Filter works better when there is preponderance of the genuine tweets preceding the fake ones because of its recursive dependence on the preceding state. mCUSUM performs better in all other cases since it is post-fact resulting in a better estimate for the expected score. mCUSUM is almost 100% accurate when the fake tweets are all inserted at once for reasons explained earlier.

For the mCUSUM threshold $\tau$, we used the range (0, 25) with 0.25 increments to find the optimum permutation of threshold $\tau$ and $\omega$ which maximizes AUC, the Area Under the Receiver Operating Characteristic (ROC) curve. However, experiments showed that the optimum threshold is quite close to 0. For sensitivity constant $\omega$, we used the range (0, 1) with 0.05 increments, covering most possible permutations of values in nested loops.

Similarly for the Kalman Filter, we used three nested loops for the offset $\tau$, and presumed constants, $R$, and $Q$ to find the permutation that yields the best AUC. Here too, the optimum $\tau$ was close to or often 0, as we saw in the plots as well.

The number of true positives influences the performance more than the other measures in some scenarios. Scenario 5 particularly, greatly depends on the accuracy of the confusion matrix. AUC can be 0 even if the indices are off by 1, demonstrating the sensitivity of the constants.

For mCUSUM, we tried computing the $\mu_0$ in the loop considering only the microblogs posted till then, like is done for basic CUSUM, instead of at the beginning, but the performance in terms of AUC values, was not as good as when we used the mCUSUM logic. Same results if $\mu_0$ is computed for just the first few tweets instead of for all. A mean representative of all the scores, post-fact is apparently a better measure for the expected sentiment to measure and accumulate the differences than the mean of just the first few tweets or even the tweets posted till then. This limits the applicability of the mCUSUM algorithm for realtime / online detection. We overcome this limitation in the framework discussed in the section on "The Proposed Comprehensive Anomaly Detection".

#### 4.4.2. False positives analysis

The algorithms are impacted by false positives as illustrated in Fig. 27. The Y-axis plots the fraction $\frac{FalsePositives}{TotalTweets}$ for each scenario given on the X-axis. For the first two scenarios, mCUSUM does not result in any false positives, whereas the Kalman Filter produces a substantial number of them. The reason is as follows. The Kalman Filter recursively depends on the previous state to detect changes. In the first two scenarios, however, there is no genuine post to establish accurate "previous state" for Kalman Filter, which makes it ineffective. On the other hand, as the mCUSUM uses the average of overall sentiment score as the mean for change detection, as long as the majority of the posts are real, it can perform accurate change detection. The recall rate for both the algorithms in all the scenarios is quite high.

The mCUSUM false positive rate for scenarios 3, 4, and 11 is particularly high because of the frequent fluctuations in the sentiment score. In all the three scenarios, fake tweets are posted at very short intervals. In scenarios 3 and 4, the interval is 1 - fake tweets alternate with the genuine ones. The more frequent the fluctuations, the more confused is the CUSUM algorithm. The first two scenarios form one pattern that is explained in the previous paragraph, where the attack happens in succession. The frequent fluctuation pattern we saw in scenarios 3, 4, and 11 is the other. All the remaining scenarios are in between these two patterns, so the false positives vary accordingly for both the mCUSUM and the Kalman Filter.

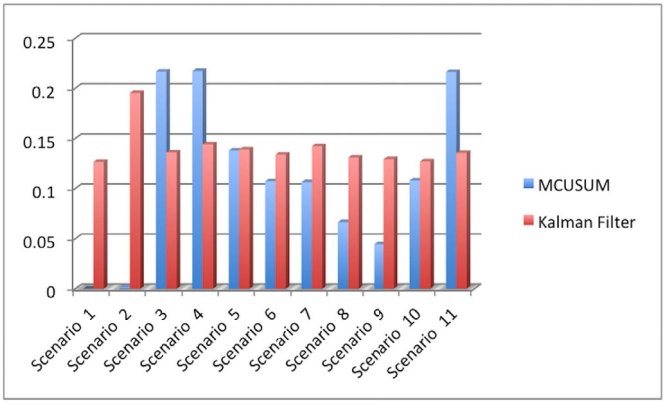


**Fig. 27.** False Positives as a fraction of the total number of tweets for each scenario plotted on the Y-axis.

## 5. The Proposed Comprehensive Anomaly Detection

From the above discussion, on one hand, it can be seen that the Kalman Filter suffers from false positives in certain scenarios. This is particularly true for the common scenarios 1 and 2, as the results in Fig. 27 show. On the other hand, the mCUSUM cannot be applied in realtime. This calls for an improved solution that we will describe in this section.

We explained in the mCUSUM section that $\mu_0$ is the average of all sentiment scores till then. Due to this limitation of the mCUSUM algorithm requiring the $\mu_0$ value to be computed on the tweets post-fact, the framework presented in the earlier sections can be used only offline. However, it is preferable if the attacks are detected in realtime to curtail any resulting damage. We address these limitations of the first framework by proposing a second framework as illustrated in Fig. 28.

Using the training data, parameters, such as $\tau$, $\omega$ for mCUSUM and $\tau$, $Q$, and $R$ for the Discrete Kalman Filter algorithms are first computed offline. The parameters are then fed into the corresponding instances of the algorithms which run on the live microblogs. The microblogs are first processed by the Discrete Kalman Filter and if the algorithm flags possible influence attacks often, mCUSUM gets invoked. It runs on the past few hundreds of the microblog posts to determine if an attack is indeed in progress and alerts accordingly.

The reason for using an ensemble of the two algorithms is for one to overcome the limitations of the other. Kalman Filter can be applied realtime, but is impacted by false positives in certain scenarios. mCUSUM is less impacted by false positives for the common scenarios but can work only on offline data for the reason discussed above. Cross-checking the realtime results from Kalman Filter with those from running mCUSUM on immediate past offline data will therefore result in

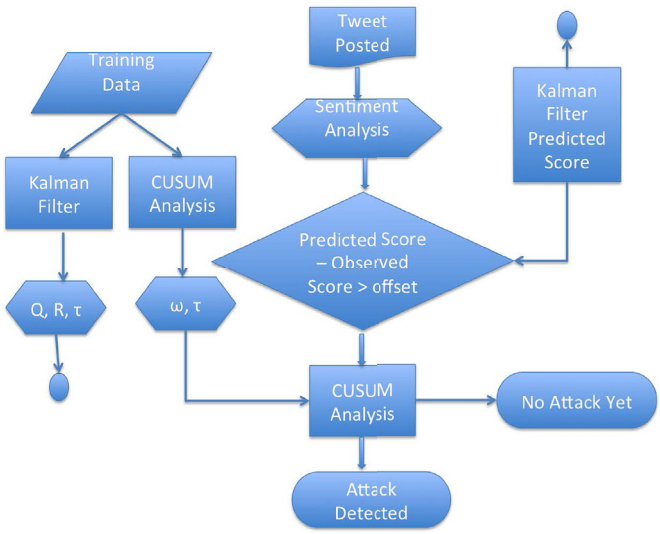


**Fig. 28.** Framework for detecting microblog attacks on OSNs.

better accuracy, when it comes to realtime attack detection. The second framework shown in Fig. 28 proposes this arrangement.

## 6. Discussion

To the best of our knowledge, we did not find similar work using the same or similar Change Detection techniques applied to sentiment scores of tweets to identify injected attacks. As mentioned earlier, solutions proposed by other researchers using other techniques, such as those presented in Vosoughi (2015) specifically were successful in identifying around 70% rumors correctly, as compared to the results from the solution presented in this paper. Also, in the above sections, we compared the results from the mCUSUM and Kalman Filter algorithms and proposed a framework that purports to workaround the drawbacks of one with the advantages of the other.

This work successfully demonstrated how change detection techniques can be used to identify influence shifts on OSNs, using anomaly detection algorithms modified for the purpose. The results confirmed our hypothesis so as to continue research in the direction. The experimental settings comprehensively cover various real-world scenarios. We used statistical techniques to uncover the patterns of adverse influence that closely follow the real-world scenarios. However, there are some limitations as discussed below.

It must be noted that the purpose of an injected influence hacker attack is to change the public sentiment on a topic. It is not an assumption, but a basic premise, the ground truth, that the attack corresponds to a change in sentiment. The only effect of a successful injected influence attack is a change in sentiment. It may be true that not all sentiment changes are a result of an attack. But it is rare that public discourse shifts significantly without a deliberate attack to inject influence.

That is the reason we identified and simulated several scenarios to see if the techniques can recognize the attacks and evaluated the accuracy for each scenario. Our results have shown that the attack could be identified with a significant rate of accuracy. It also showed that there are quite a few false positives, those which were not injected, but were a result of other factors like hot mic etc. The false positives are duly tabulated and incorporated into the evaluation of our algorithms.

It is well known that most estimators and predictors work well on Gaussian distributions. This holds for mCUSUM and Kalman Filter. However, as mentioned earlier, Gaussian assumption is not mandatory for CUSUM (Philips) and for Kalman Filter as well. But Gaussian distributions do help improve the performance in both cases.

As can be seen from the mCUSUM Algorithm #1, the mean is computed on all the sentiment scores of the microblogs so far. That means the mCUSUM algorithm works only post-fact. Only after we analyze the sentiments of all the microblogs do we know which one of them could possibly be part of an attack. Kalman Filter can fare better in that respect as it can immediately predict the sentiment score of the next microblog and if the actual score differs from this by a significant offset, it can be considered as manipulating the influence.

It must also be noted that we do not distinguish between the microbloggers. Typically, multiple malicious accounts are controlled by one attacker. They share the same attack goal as to increase or decrease the sentiment of the microblog. Computing the correlations between the accounts may further help in the detection process. Also, for our experiments, we focused only on negative sentiment influence because many studies, such as (Ansolabehere et al., 1999; Baumeister et al., 2001) have confirmed that negativity has the most impact on campaigns. The human mind has greater sensitivity to unpleasant news. The work can easily be extended to cover positive sentiment.

The results from the framework we proposed can be further confirmed by using the framework in conjunction with some of the other techniques we discussed in the related work section. Beyond this, it may not be reasonably possible to really know if a tweet has been posted with a malicious intent. More details and discussion on the topic are presented in the first author's book (Pendyala, 2018).

## 7. Conclusion and future directions

A malicious scheme that has the potential to alter the outcome of a country's presidential election is serious problem. Through this work, we proposed a few ways in which such schemes can be detected, experimentally evaluated the methods, and achieved significantly accurate results. As highlighted in the Related Work section, Misinformation Containment is proven to be NP-hard. We cannot solve the problems in this category completely accurately. There is no definitive way to determine if an opinion shift on microblogging sites is entirely attributable to a malicious attack. But the algorithms we used to the solve the problem seem to be reasonably accurate at making such a determination. The accuracy of determining a deliberate injected attack can be further improved by correlating the results from our experiments with true happenings, such as described in the news media or other more reliable sources. The ideas presented in this paper can be further strengthened by using them in conjunction with technologies such as blockchain.

There are a few other ways this work itself can be enhanced further. First and foremost is to do the experiments live, probably on an exclusive students' internal microblogging website, creating one if it doesn't already exist and study how the injected microblogs influence the opinion of others. We can then see if the algorithms presented here can actually detect the injected microblogs.

Microbloggers' correlation analysis can help achieve better accuracy rates. Simple Euclidean distance can be used for user correlation analysis. However, the condition for incorporating correlation analysis of microbloggers is that the users involved in one topic are also involved in other topics. This condition may not be true for this specific case if the attacker's only goal is the manipulation of one specific topic. There is some scope for improvement in generating the semantic scores as well.

While surveying the existing literature in Section 2, we have highlighted how this work can be used in conjunction with other techniques. Those are some of the enhancements that can be taken up for future work.

A future project based on this one is to deploy the change detection techniques on news articles, text transcripts of video and audio news, and news media in general. It will show how much media can influence crucial events, such as a Presidential election. Such a project can be particularly useful given that allegations of a corrupt, paid media are growing day by day in large democracies and developing countries like India.

This paper is hoped to open up plethora of ideas in the direction of evolving an entirely truthful World Wide Web, eventually.

## Conflicts of interest

There are no conflicts of interest.

## Acknowledgment

The authors are grateful to Santa Clara University for funding the Open Access publication fee for this article.

## References

Ansolabehere, S., Iyengar, S., Crigler, A.N., Holbrook, T.M., Huckfeldt, R., Sprague, J., 1999. Going negative. In: How Political Advertisements Shrink & Polarize the Electorate.

Baumeister, R.F., Bratslavsky, E., Finkenauer, C., Vohs, K.D., 2001. Bad is stronger than good. Rev. Gen. Psychol. 5 (4), 323.

Bessi, A., 2017. On the statistical properties of viral misinformation in online social media. Phys. A Stat. Mech. Appl. 459–470.

Bird, S., Klein, E., Loper, E., 2009. Natural Language Processing with Python: Analyzing

Text with the Natural Language Toolkit. O'Reilly Media, Inc.
Budak, C., Agrawal, D., El Abbadi, A., 2011. Limiting the spread of misinformation in social networks. In: Proceedings of the 20th International Conference on World Wide Web. ACM, pp. 665–674.
Burns, A., Eltham, B., 2009. Twitter Free iran: an Evaluation of Twitter's Role in Public Diplomacy and Information Operations in iran's 2009 Election Crisis. Retrieved from: http://vuir.vu.edu.au/15230/1/CPRF09BurnsEltham.pdf.
Castillo, C., Mendoza, M., Poblete, B., 2011. Information credibility on twitter. In: Proceedings of the 20th International Conference on World Wide Web. ACM, pp. 675–684.
Chew, C., Eysenbach, G., 2010. Pandemics in the age of twitter: content analysis of tweets during the 2009 h1n1 outbreak. PLoS One 5 (11), e14118.
Choy, M., Cheong, M.L., Laik, M.N., Shung, K.P., 2011. A Sentiment Analysis of singapore Presidential Election 2011 Using Twitter Data with Census Correction. arXiv preprint arXiv:1108.5520. .
Cogburn, Derrick L., Espinoza-Vasquez, Fatima K., 2011. From networked nominee to networked nation: examining the impact of Web 2.0 and social media on political participation and civic engagement in the 2008 Obama campaign. J. Polit. Market. 10 (1–2), 189–213.
Hall, Wendy, Tinati, Ramine, Jennings, Will, 2018. From Brexit to Trump: social mediaś role in democracy. Computer 51 (1), 18–27.
Hamidian, S., Diab, M.T., 2016. Rumor identification and belief investigation on twitter. In: Proceedings of NAACL-HLT, pp. 3–8.
Hernandez-Suarez, Aldo, Sanchez-Perez, Gabriel, Toscano-Medina, Karina, Martinez-Hernandez, Victor, Perez-Meana, Hector, Olivares-Mercado, Jesus, Sanchez, Victor, 2018. Social sentiment sensor in twitter for predicting cyber-attacks using l1 regularization. Sensors 18 (5).
Hutto, C.J., Gilbert, E., 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Eighth International AAAI Conference on Weblogs and Social Media.
Jain, S., Sharma, V., Kaushal, R., 2015–2020. Towards automated real-time detection of misinformation on twitter. In: Advances in Computing, Communications and Informatics (ICACCI), IEEE International Conference on, pp. 2016.
Jin, F., Dougherty, E., Saraf, P., Cao, Y., Ramakrishnan, N., 2013. Epidemiological modeling of news and rumors on twitter. In: Proceedings of the 7th Workshop on Social Network Mining and Analysis. ACM.
Jin, Fang, Wang, Wei, Zhao, Liang, Dougherty, Edward, Cao, Yang, Lu, Chang-Tien, Ramakrishnan, Naren, 2014a. Misinformation propagation in the age of twitter. Computer 47 (12), 90–94.
Jin, F., Wang, W., Zhao, L., Dougherty, E., Cao, Y., Lu, C.T., Ramakrishnan, N., 2014b. Misinformation propagation in the age of twitter. Computer 47 (12), 90–94.
Kumar, K.K., Geethakumari, G., 2014. Detecting misinformation in online social networks using cognitive psychology. Human-centric Comput. Inf. Sci. 4 (1), 1.
Kwon, S., Cha, M., Jung, K., 2017. Rumor detection over varying time windows. PLoS One 12 (1) e0168344.
Laquintano, Timothy, Vee, Annette, 2017. How automated writing systems affect the circulation of political information online. Lit. Compos. Stud. 5 (2), 43–62.
Liu, B., 2012. Sentiment analysis and opinion mining. Synth. Lect. Human Lang. Technol. 5 (1), 1–167.
Liu, Y., Sun, Y., 2010. Anomaly detection in feedback-based reputation systems through temporal and correlation analysis. In: Social Computing (SocialCom), IEEE Second International Conference on, pp. 65–72.
Liu, X., Nourbakhsh, A., Li, Q., Fang, R., Shah, S., 2015. Real-time rumor debunking on twitter. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM, pp. 1867–1870.
Nguyen, N.P., Yan, G., Thai, M.T., Eidenbenz, S., 2012. Containment of misinformation spread in online social networks. In: Proceedings of the 4th Annual ACM Web Science Conference. ACM, pp. 213–222.
Nielsen, F., 2011. A New Anew: Evaluation of a Word List for Sentiment Analysis in Microblogs. arXiv preprint arXiv:1103.2903. .
Page, Ewan S., 1954. Continuous inspection schemes. Biometrika 41 (1/2), 100–115.
Pak, A., Paroubek, P., 2010. Twitter as a corpus for sentiment analysis and opinion mining. LREc 10.
Pendyala, Vishnu, 2018. Veracity of Big Data: Machine Learning and Other Approaches to Verifying Truthfulness. Springer.
Pendyala, V.S., Figueira, S., 2015. Towards a truthful world wide web from a humanitarian perspective. In: Global Humanitarian Technology Conference (GHTC), IEEE International Conference on.
Pendyala, V.S., Figueira, S., 2017. Automated medical diagnosis from clinical data. In: Third International Conference on Big Data Computing Service and Applications.
Pendyala, V.S., Fang, Y., Holliday, J., Zalzala, A., 2014. A text mining approach to automated healthcare for the masses. In: Global Humanitarian Technology Conference (GHTC), IEEE International Conference on.
Thomas K. Philips. Monitoring active portfolios: The cusum approach. http://www.northinfo.com/documents/144.pdf.
Qazvinian, V., Rosengren, E., Radev, D.R., Mei, Q., 2011. Rumor has it: identifying misinformation in microblogs. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, pp. 1589–1599.
Ritter, A., Wright, E., Casey, W., Mitchell, T., 2015. Weakly supervised extraction of computer security events from twitter. In: Proceedings of the 24th International Conference on World Wide Web. ACM, pp. 896–905.
Singh, M., Bansal, D., Sofat, S., 2014. Detecting malicious users in twitter using classifiers. In: Proceedings of the 7th International Conference on Security of Information and Networks. ACM.
Starbird, K., Maddock, J., Orand, M., Achterman, P., Mason, R.M., 2014. Rumors, false flags and digital vigilantes: misinformation on twitter after the 2013 boston marathon bombing. In: iConference Proceedings.
Takahashi, T., Igata, N., 2012. Rumor detection on twitter. In: Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), Joint 6th International Conference on, pp. 452–457.
Tumasjan, A., Sprenger, T.O., Sandner, P.G., Welpe, I.M., 2010. Predicting elections with twitter: what 140 characters reveal about political sentiment. In: ICWSM, vol. 10. pp. 178–185.
Vosoughi, S., 2015. Automatic Detection and Verification of Rumors on Twitter (Doctoral Dissertation). .
Welch, G., Bishop, G., 2001. An introduction to the kalman filter. In: SIGGRAPH Course Notes. ACM, Los Angeles, CA Course 8. ACM.
Wu, K., Yang, S., Zhu, K.Q., 2015. False rumors detection on sina weibo by propagation structures. In: 31st International Conference on Data Engineering, pp. 651–662.
Yang, F., Liu, Y., Yu, X., Yang, M., 2012. Automatic detection of rumor on sina weibo. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics. ACM.