The Journal of Supercomputing (2019) 75:6085–6105
https://doi.org/10.1007/s11227-019-02948-w

An enhanced graph-based semi-supervised learning algorithm to detect fake users on Twitter

M. BalaAnand[1] · N. Karthikeyan[2] · S. Karthik[3] · R. Varatharajan[4] · Gunasekaran Manogaran[5] · C. B. Sivaparthipan[3]

Published online: 30 July 2019

Abstract
Over the the past decade, social networking services (SNS) have proliferated on the web. The nature of such sites makes identity deception easy, providing a fast means for creating and managing identities, and then connecting with and deceiving others. Fake users are those accounts specifically created for purposes such as stalking or abuse of another user, for slander, or for marketing. The current system for detecting deception depends on behavioral, non-behavioral and user-generated content (UGC) information gathered from users. Although these methods have high detection accuracy, they cannot be implemented in databases with massive volumes of data. To address this issue, this paper proposes an enhanced graph-based semi-supervised learning algorithm (EGSLA) to detect fake users from a large volume of Twitter data. The proposed method encompasses four modules: data collection, feature extraction, classification and decision making. Data collected from Twitter using Scrapy is utilized for the evaluation. The performance of the proposed algorithm is tested with existing game theory, *k*-nearest neighbor (KNN), support vector machine (SVM) and decision tree techniques. The results show that the proposed EGSLA algorithm achieves 90.3% accuracy in spotting fake users.

Keywords Fake user detection · Identity deception · Sock puppets · Semi-supervised learning algorithm

1 Introduction

Online social networks (OSNs) have become one of the most important ways for people to interact with their friends, to grow a business or to obtain information. By some estimates, up to 67% of people aged 18–29 exploit social networking at some level, and around 22% of the population overall uses social network websites for

✉ M. BalaAnand
balavdy@gmail.com

Extended author information available on the last page of the article

some reason. These days, social networks such as Twitter and Facebook are driving different sorts of interaction, collaboration, exchange and dialogue [1].

Typically, individuals tend to use different OSNs concurrently. For example, an individual uses Facebook to connect with his friends, uses Twitter to post news, and uses Foursquare for location-centered social activities. As expected, his social activities and associations are scattered on some networks, so his online information on a solitary website is frequently inadequate. If precisely incorporate these sites, can fabricate that individual's information to be better in addition to more complete to enhance online services, for example, community discovery, recommendation [2]. Commonly, in a social network, users post their personal traits on their personal pages, for example, their colleges, majors and interests. This profile data can be used as a reason for grouping users, sharing content, and suggesting friends. However, in reality, not every user on a social network is real [3].

Fake users are defined as any account that consistently involves automation over an observed period, such as the use of the Twitter API or other third-party tools, or performing actions such as automated likes, tweets, and retweets. Fake users include marketing bots, social bots, stalker bots and malicious bots [4]. Every bot has a unique purpose. Marketing bots tweet about a specific product or company in a predefined design. A social bot retweets love likes and follow in light of certain key words (e.g. smile). A stalker bot spins around a user or two. Malicious bots spread spam links [5]. A tweet is an original status and not a retweet, a retweet is a tweet with "RT" in the text, and status is either a tweet or a retweet. Content on Twitter is limited to whatever is contained in a tweet: text, URL, picture, and video [6].

The fundamental difficulty in detecting fake users is that they are rapidly refining their spam strategies to stay ahead of advances in detection measures [7]. For methods that use essential features focused upon user profiles and message content, fake users can evade detection by purchasing followers or using tools to post messages with a similar meaning but with different words [8, 9]. These bots tend to be interconnected, surrounding account groups. Supporter accounts help fake users evade detection by increasing their numbers of followers. In any case, perceived as social business people are a small percentage of real users, and they will take after back any person who follows them, with the specific goal of manufacturing their notoriety. There are a set number of edges keeping up complementary social connections between fake users and real users, but the confirmation that real users take after the fake user more than anticipated was found [10].

Fake users registered on social networks are able to perform a variety of actions (depending on the specific social network), as follows: log onto the social network home page and access a personal page, modify personal settings using the graphical interface of the social network by providing personal information (full name, virtual communities of interest, educational institutions, list of employers, vacation destinations, areas of interest, recreational activities, and musical and other interests); change (create, modify or delete) current "status" bar values on personal page (typically contains current activity, disposition or geolocation indicated by the user or the user's device); post text, photo or audio materials on the personal page on social network home page; frame or cut off friendly relations (both one-way and two-way—usually confirmed by the user); send personal messages to a specific social network user, which is undetectable by other users; comment on text, photo or audio materials published by other users by distributing instant messages immediately underneath them; express approval or disapproval regarding other users' posts or comments by marking them positive or negative using the graphical interface of the social network community; finish the operation in the social network as a registered user at any point, breaking the authorization session and using the "log out" soft key in the social network graphical interface.

Consider that there is access to open data aggregated on random social network users, including personal details available on their social network accounts, registration date and time, actions performed at certain time points, a set of distributed text messages in open access, a set of open, friendly relationships developed by social network users and stored in a social database [11]. The relationship between users can be represented in the form of graph arcs connecting the corresponding nodes of the graph, and so these relations can similarly be described as a vertex matrix. Formal markers can also indicate the presence of the social bot subset in the social networks. Big data concerns assume a key role in the accurate analysis of data from OSNs. The analysis of data from OSNs can aid communications media in various respects [12]. Some key challenges with respect to big data analysis can be highlighted using a few examples. Algorithms that can be used to process big data include ensemble learning in order to improve performance by reducing computational cost, model unpredictability methods to optimize time complexity, local learning frameworks to reduce computational complexity, semi-parametric approximations for global models, deep learning to increase chip processing ability and further lower computing hardware cost, big data processing with MapReduce, GraphLab for batch processing, and Storm and SAMOA for stream processing [13, 23].

In this paper, we introduce an enhanced graph-based semi-supervised learning algorithm (EGSLA) to detect fake users by examining the activity of users over a broadened time span. Intuitionally, but various spammers can make their accounts appear like real user accounts at some fixed time points to evade detection, it is not possible for them to control the dynamic changing system of features over an extended time period because of the high cost. Scalable streaming, which includes the batch and stream processing, is a real-time model for scalable processing of social data for classification along with prediction. Batch processing is used to train machine learning (ML) models, which is not sufficient for real-time applications. The EGSLA model performs classification, makes predictions and gives recommendations.

The remainder of this paper is structured as follows: Sect. 2 covers related works on fake user detection in OSNs; Sect. 3 covers the proposed method (EGSLA); Sect. 4 presents the results and analysis of the proposed EGSLA; and Sect. 5 concludes the paper.

2 Related works

Boshmaf et al. [14] introduced Integro, a scalable defense system that helps OSN administrators to detect fake accounts by means of a large-scale user ranking system. OSNs today are confronted with the problem of identifying fake accounts under extremely adversarial conditions. The issue is becoming even more challenging as these accounts have become increasingly sophisticated and adept at concealing their activity with patterns that look like real user behavior. The authors showed that SybilRank, the state of the art in fake account detection, was ineffective when the fake users infiltrated the target OSN by befriending various real users. Integro, however, exhibited greater resilience to this effect by making novel use of information about benign victim accounts that befriended fake users. The authors implemented Integro on the standard data processing platforms Mahout and Giraph, which are scalable and easily deployed in present-day server farms.

Escalante et al. [15] proposed a methodology for profile-based detection of deception along with aggressiveness in written text. Profile-based representations utilize class term occurrence in order to deduce a non-sparse, low-dimensional, discriminative representation for documents, where profiles can be further divided into subprofiles or subclasses. Because these representations can be estimated even when only a single term is available, they are suitable for analysis where documents have missing or incomplete information. In light of this advantage, the authors developed early recognition methods based on profile representations and assessed them in data sets for distinguishing deception and aggressiveness, where the use of a minimum amount of information was critical for prevention. The experimental results showed that these representations could be used in a preventive way to perform detection tasks. When classifying full-length documents, profile-based representations did not appear to contribute to improved recognition, although in many cases they improved the performance of alternative methods, achieving superior performance. In early recognition, profile-based representations enhanced the performance of the reference method, chiefly sexual predator detection and Kaggle data sets [20]. In these instances, utilizing profile-based representations enhanced both the jump-start and classification performance when reading a small amount of data, which is an especially desirable feature for any early detection method. Subprofile-based representations performed better than profile-based representations in both sexual predator detection and aggressive text identification. Profile and subprofile representations were sensitive to the amount of data available in the training data set, in terms of both sparseness and number of samples. This issue affects subprofile-based representations significantly. Aggressive text classification is a more complex problem than sexual predator detection for ML techniques. Among the classifiers that were considered, neural networks demonstrated the most consistent performance across data sets, achieving state-of-the-art performance. However, the early recognition capacity of this strategy was limited.

Tsikerdekis et al. [16] noted that to use these systems logically, one needed to consider three basic factors that affect their capability: the number of users on a platform, the innovation framework and the information speed. Their work arranged detection methods based on the approach and identified components that continuously structures will influence the adequacy and capability of these procedures [21]. Advancements were proposed that can control computational costs. Facilitate real-time personality trickery recognition made an order of existing duplicity discovery approaches and proposed factors and moreover improvements that should be considered for a continuous utilization of these systems in web-based social networking.

These proposals require a reevaluation of identification approaches and their potential application to online networking [22]. Given the rapid advances in online networking together with increasing data speed, it was likely that innovation won't extra the day for these identification approaches but rather creative new ways that will make them handy went for real-time applications.

Kuruvilla et al. [17] proposed a system for identifying multiple accounts owned by a single user based upon verbal and nonverbal behavior. This system identified multiple accounts maintained by a single user by utilizing both the user's verbal and nonverbal behavior. It distinguished nonverbal behavior by utilizing ML calculations. Wikipedia has page revision history, which contains the actions of every user, for example, posts made by users, posts deleted/edited by users, and the time taken to create a modification. Every user has separate correction log documents. It recognized the fake users based on the execution time of user activity and individual writing style. This system also had high detection and prediction accuracy. However, it did not think about the parameters for the users protection.

Gera and Singh [18] proposed a system for identifying the authenticity of each survey posted on product review sites. They demonstrated a parameterized approach for recognizing questions, review spammers and their groups considering geological measurements alongside systems administration parameters [24]. The proposed hostile review spam detection framework combines a two-stage methodology for detecting opinion spammers and their groups [22, 25]. In future work, the proposed parameterized approach will be completed to produce results with good accuracy. It was found that this method would similarly diminish the spam impact, so to speak, and, similar to this assistance in making the better appraising quality for a thing which will aid many users in making better purchase decisions.

Jiang et al. [19] proposed QuickSquad, a graph computing system on a single server that is specific to social network graph structure optimization. The authors first divided the graph structure data into heavy and light sets according to the out-degree of vertices. They then stored them with different formats, processed them with the appropriate edge-based and vertex-based updating in a two-phase processing model, applied two selective scheduler strategies of different levels, and provided four cache priorities when the memory was insufficient to cache all data. They then implemented two detection methods, dSybilRank and dCOLOR. In the experimental evaluation, this approach was compared with existing approaches.

3 Proposed methodology for fake user detection in OSN

The proposed strategy is an EGSLA that uses graph-based classification techniques to distinguish fake users from a pool of Twitter users. Here, we outline a system to consequently gather user-generated content (UGC) from Twitter and to store them in the database. The composed data are processed to extract the features. The features are utilized in the EGSLA classification module and predicated on the weighted graph, and the data are labeled. Based on the label, the believability of the user account is determined. This innovative fake user detection method comprises three steps, as enumerated below and illustrated in Fig. 1.

1. Data collection
    - Python web-scraping framework
2. Feature extraction
    - Fraction of retweets
    - Standard tweet length
    - Fraction of URLs
    - Average time between tweets:
3. Classification and decision making
    - EGSLA algorithm

Graphs are mathematical structures utilized for the modeling of pairwise relationships between objects. Graph models can be classified into directed graphs and undirected graphs. An undirected graph implies that there is no distinction between two vertices related to every edge. For instance, if each vertex signifies a man at a social event, an edge between two individuals implies that they shake hands, and after that the graph will be undirected. Despite what might be expected, a directed graph incorporates directional edges, and the direction is shown by arrows. For instance, if each vertex signifies a site page, and edges signify a hyperlink starting with one site page and then the next, at that point the graph will be directed. The most commonly utilized graphs in computer science are weighted graphs. The fundamental components of weighted graphs are vertex, edge and weight. In its mathematical representation, a graph is an ordered pair $G = (V, E, W)$, where V is a set of vertices or points or nodes, E is a collection of edges or lines or arcs that are two-component subsets of V, and W is a set of weights of the vertices or the edges.

A graph-based model is utilized for modeling user behavior. Typically, the nodes of such a graph are associated with individuals, and the edges connecting the nodes are associated with the type of communication or connection between the general population. All fake users display the same example. These parameters can be restrictively joined to detect the likeness between users.

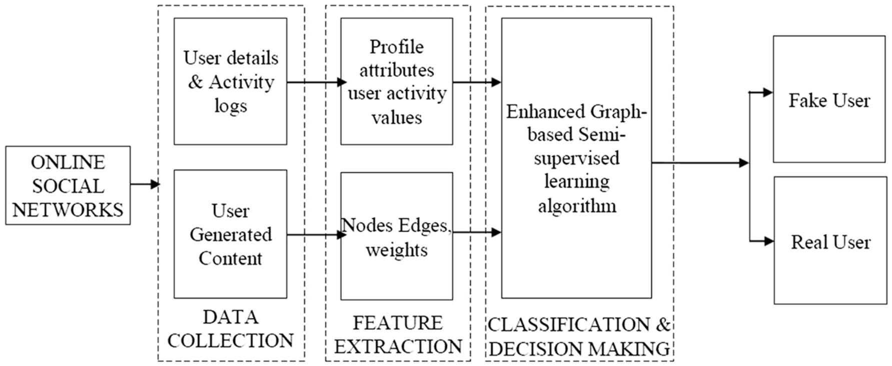
ONLINE SOCIAL NETWORKS
User details & Activity logs
User Generated Content
DATA COLLECTION
Profile attributes user activity values
Nodes Edges, weights
FEATURE EXTRACTION
Enhanced Graph-based Semi-supervised learning algorithm
CLASSIFICATION & DECISION MAKING
Fake User
Real User

Fig. 1 Proposed fake user detection system

- Fake users post the same content in multiple accounts. The group of instant messages posted by the user can totally involve messages which were taken from various users. Accordingly, for any message of the *i*th user, the unclear message shows in the social database that was distributed before by some *j*th user. The comparative way participation work aimed at social bot subset that relies upon weighted factors can be assembled.
- Fake users perform activities at regular intervals. Different users perform the same activities at regular predefined time intervals.
- Fake user accounts have similar patterns. This is a typical pattern which appropriates for every individual data of the *i*th and *j*th users and exist after that it is sensible to assume the registration of *i*th and *j*th virtual users to be mostly administered by sophisticated software. A typical example that is reasonable for the name order of accounts ("kumar_2000", "kumar_2001", "kumar_2002", etc.) as though which are prepared by a unique algorithm. Utilizing the regular pattern naming causes the authorized person to get the job statistics is a rule.
- Fake users have either links or media content in all the posts. Likewise, they utilize trending hashtags. This is done to expand their tweet reach. However, the content of the post and the hashtags or multimedia are not really related.

3.1 Data collection

The initial step is gathering the necessary data for the analysis. Here, 2 months of Twitter data have been collected, and as many tweets as possible are collected from the Twitter handles that tweet with links and trending topics. It is assumed that fake users tweet with links in every tweet. In addition, fake users like to tweet with trending hashtags to increase their visibility. Trending topics and external links are the fundamental data sources. In this approach, the data are collected using the Python web-scraping framework. Web scraping is an automated method utilized to extract large quantities of data from websites. The data on the websites are unstructured. Web scraping aids in collecting this unstructured data and storing it in a structured form. A crawler is implemented in the Python Scrapy framework to download the above-mentioned parameters in Twitter. The raw data gathered by the Scrapy system is stored in a Neo4j instance. Neo4J is a graph database that stores nodes and the relationships between nodes. Neo4J is the most popular graph database according to DB-Engines Ranking, and the 22nd most popular database overall. Neo4J has several types of graphs, one of which is a directed graph, where nodes and relationships by default are not bidirectional. Twitter relationships are of this type, where a user can follow certain profiles in this social network without them following him. It can scale limitlessly and can efficiently perform graph calculations (for example, "discover the nodes that are linked with a node that is linked with another node through a specific relationship"). Finally, the Twitter data collection consists of user id, status, followers list, following information, posting time, retweet status, tweet length and time between tweets.

The data are preprocessed utilizing Apache Spark to format and standardize the raw data to create data sets. Apache Spark is a fast computing technology; it increases the processing speed of the application and also supports multiple languages. The preprocessor comprises two classes, one for connecting with the Neo4j instance and gathering the data centered upon a cipher query, and a subsequent class that performs all the formatting and normalizing. The preprocessed data contain the graph of user id, status, follower list and following information for the users. The data were gathered periodically at 1-h intervals to extract all trending data from Twitter. The crawler utilized the breadth-first search algorithm to crawl Twitter. A total of 21,473 users and 2,915,147 tweets were ultimately acquired.

3.2 Feature extraction

This section exhibits the formulation for all of the features employed for fake user detection. A user typically posts his UGC on different accounts at similar time periods. Such temporal distribution reveals his online action pattern. For instance, some users like to post UGC toward the start of the day, while other users conduct their activity at night. Such users' online patterns are very helpful for user identification. For a user, this can extract the posting time from his public UGC and acquire a succession of posting times. Similar content is found on various accounts, which demonstrates that fake users generally publish their data on numerous different accounts. This UGC contains a considerable number of similar words or synonyms. The similarity of UGC content can be indicated by the semantic similarity of words or the numbers of the same words.

3.2.1 Fraction of retweets

This measures the number of times the user distributed a retweet divided by the number of tweets the user has distributed, calculated as:

$$\text{Retweet}(u) = \frac{|\{x|x \in \text{tweets}^u, x = \text{retweet}\}|}{|\text{tweets}^u|} \tag{1}$$

Whether a tweet is a retweet or not is decided by looking at the "retweeted status" field of the tweet's data when returned through the Scrapy system. If the field contains tweet information, view it as a retweet.

3.2.2 Average tweet length

A tweet's text is limited to 280 characters, but it is possible that bots will post less than that, as they could simply be advancing a URL or a hashtag. To take this into account, the average length of the user's tweets is checked. It is the total of the characters in the whole tweets the user has distributed divided by the number of tweets the user has distributed, formally:

$$\text{Length}(u) = \frac{\sum_i^{|\text{tweets}^u|} \left|\text{tweets}_i^u\right|}{|\text{tweets}^u|} \tag{2}$$

where $\left|\text{tweets}_i^u\right|$ is the length of the user u's ith tweet, estimated by the quantity of characters in the tweet.

3.2.3 Fraction of URLs

This measures the number of times the user published a tweet containing a Uniform Resource Locator (URL) divided by the total number of tweets the user has published:

$$\text{URL}(u) = \frac{|\{y|y \in \text{tweets}^u, y = \text{URL}\}|}{|\text{tweets}^u|} \tag{2}$$

Whether a tweet contains a URL is determined by looking at the "entities" field of the tweet returned by the application programming interface. Bots attempt to persuade genuine users to go to external sites that are operated by their controller. This component allows one to distinguish bots that are attempting to promote URLs in their tweets. The entry point of the URL is found for each tweet. It gauges the frequency with which every URL occurs in the tweet.

3.2.4 Average time between tweets

It was discovered that various bots tweet in a "bursty" nature, distributing an extensive proportion of their tweets within a short period. This lead is measured as:

$$\text{Time}(u) = \frac{1}{|\text{tweets}^u| - 1} \sum_{i-2}^{N} (t_i - t_j) \tag{3}$$

where t_i is the time stamp of the ith tweet in user u's timeline, t_j is the time stamp of the jth tweet in user u's timeline, sorted chronologically in ascending order (i.e. $t_\mathrm{i} > t_\mathrm{j}$).

The similarity of posting time can be signified as the distinction in time. Also, dole out weight w_{im}^1 to posting time t_{im}^1. For two users (u_i^1, u_k^2) can calculate the distinction between two successions of time, and obtain a time similarity matrix

$$M_{ik}^t = \{M_{mn} \times w_{im}^1 \times w_{kn}^2\} \tag{4}$$

where M_{mn} indicates the contrast between the mth posting time of user v_i^1 and the nth time of user v_k^2. When considering the distinction on the users' online behavior patterns, computed the time similarity in two unique granularities, date and time, in addition get two related matrices M_{ik}^{t1} and M_{ik}^{t2}. To signify the temporal features, extracted the time distinction distribution from M_{ik}^{t1} and M_{ik}^{t2}, and mean these features as the vector V_{ik}^t. Likewise, for two users (u_i^1, u_k^2), it can calculate the similarity between any two sets of UGC, and obtain a content similarity matrix $M_{ik}^p = \{M_{mn}\}$.

3.3 Classification and decision making

The task of distinguishing true from false stories is frequently seen as a binary classification task, although, depending upon the application, it may be perceived as a task evaluating the probability of trickiness. The graph-based approach depends on user activity and their account details. This graph-based learning provides a helpful approach for modeling data on classification problems. The reason for choosing a graph-based approach is that some data sets are naturally represented by a graph, which represents heterogeneous data uniformly, and is easily parallelizable, scalable to large data and effective in practice. One can classify each account as fake or real by recognizing the unique features of the account. A test data set is created with plausible positive and negative sentiment scores. With regard to the sentiment score range, for example, if it ranges from − 2 to 2, and − 2 is a negative score, then − 1 is a negative and 1 is a positive score. Keeping in mind the end goal of determining the similarity, 800 arbitrary users' tweets about a trending hashtag are utilized. The extracted features are trained using an EGSLA. It uses labeled and unlabeled samples to train itself, which circumvents the issue of costly group training samples with labels and significantly helps the measure of accessible training samples.

Assuming a set of l labeled samples and u unlabeled samples are provided:

$$\{(x_i, y_j)\}_{i=1}^{l} \text{ Labeled samples} \tag{5}$$

$$\{(x_i)\}_{i=l+1}^{l+u} \text{ Unlabeled samples} \tag{6}$$

$$X_{\text{test}} = \{x_i\}_{i=1}^{l} \text{ Test samples} \tag{7}$$

A binary classification issue is analyzed here, for instance, where

$$y_i \in \{-1, 1\} \text{ Class label} \tag{8}$$

Consequently, the label allocation on the sample set is described by the vector

$$y_i \in \{-1, 1\}^l \tag{9}$$

The learning algorithm can be depicted by a function f. Indicate $F \in R^n$ to be the vector of f's values on the $l + u$ points. At that point, this F contains two sections: $F_l \in R^l$ for the l labeled samples, and $F_u \in R^u$ for the u unlabeled samples.

The learning algorithm would then be transformed into the following optimization issue:

$$\min F^T SF + k_1 L(F_l, y_l) + K_2 \chi(f) \tag{10}$$

where K_1 and K_2 are regularization parameters. S is a matrix that catches the smoothness of the anticipated samples as for the data's manifold structure. By limiting the initial term, the smoothness is guaranteed. Regularization parameters keep the model vector growing arbitrarily for negligible decreases in the error, and are selected based upon the real application and for fitting interpretation of the data structure. The characteristics of regularization parameters are to reduce the error by fitting a function appropriately on the given training set and to avoid overfitting. Regularization functions vary in high density, which is given as

$$\langle f, lf \rangle = \langle f, (D - K)f \rangle = \sum_{i,j=1}^{n} w_{ij}\left(f_i - f_j\right)^2 \tag{11}$$

where l denotes the labeled samples and w_{ij} denotes the weight function.

Various techniques can be employed for the selection of S and χ. For instance, the graph Laplacian matrix (the Laplacian matrix is used to compute the number of connected components of a graph and can represent the graph structure)$S = D - K$ can be engaged for S, where K is the kernel matrix of the graph, which is expressed as

$$K = \sum_{i} \eta_i \psi_i \psi_i^T \tag{12}$$

where ψ_i's implies the eigenvectors of the graph Laplacian, $\eta_i \geq 0$ are the eigenvalues of the kernel matrix, T denotes the transverse value of eigenvector which is obtained from the related kernel function, and D is the equivalent degree matrix. Instead, the normalized graph Laplacian matrix can be engaged:

$$S = I - D^{-\frac{1}{2}} K D^{-\frac{1}{2}} \tag{13}$$

where I is the identity matrix. Once S is determined, $\chi(f)$ can be deduced by utilizing

$$\chi(f) = \left\| F_{\mathrm{u}} \right\| \tag{14}$$

The local and global consistency error function is

$$E(F) = \frac{1}{2} \sum_{i,j} w_{ij} \sum_{c} \left\| \frac{F_{ic}}{\sqrt{D_{ij}}} - \frac{F_{jc}}{\sqrt{D_{ij}}} \right\|^2 + \left(\frac{1}{\alpha} - 1 \right) \sum_{i} \sum_{c} \left\| F_{ic} - B_{ic} \right\|^2 \tag{15}$$

where c denotes the local and global consistency of the label score F. F_{ic} and F_{jc} represent the label score and B_{ic} represents the initial label assignments. The label scores are assigned for each node in the graph; the label scores differ smoothly over the graph structure. The parameter $0 \leq \alpha \leq 1$ controls the rate at which the label information is propagated around the network. To avoid computational and memory cost the operation uses the power method to iteratively update F:

$$F_{t+1} = Z^{-1}\left((1 - \alpha)B + \alpha L F_t\right) \tag{16}$$

where F_t denotes the current label score, F_{t+1} denotes the updated label score, Z^{-1} signifies a diagonal matrix used to normalize the rows of F, and B denotes the initial label assignments. The proposed EGSLA pseudo code is exhibited in Fig. 2.

```
Input: Extracted Features
Output: Detect Fake users on twitter

Begin
    Initialize $\{(x_i, y_j)\}_{i=1}^{l}$ Labeled samples, $\{(x_i)\}_{i=l+1}^{l+u}$ unlabeled samples,
        $X_{test} = \{x_i\}_{i=1}^{l}$ Test samples and $M_i$ maximum iteration.
    Set preliminary value $t = 0$.
    Compute graph Laplacian matrix $S = D - K$
    Compute normalized graph Laplacian matrix $S = I - D^{-\frac{1}{2}} K D^{-\frac{1}{2}}$
    While $(t < M_i)$ do
        Update the label score using,
            $F_{t+1} = Z^{-1}((1-\alpha)B + \alpha L F_t)$
        Set $t = t + 1$
    End while
End
```

Fig. 2 Pseudo code for EGSLA algorithm

Table 1 Overview of data set

Date accounts created	December 3, 2017
Date analysis started	December 3, 2017
Date analysis ended	February 3, 2018
Users	21,473
Tweets	2,915,147

4 Results and analysis

The proposed approach is assessed based on real posts from Twitter. Using the proposed method, 2,915,147 tweets from 21,473 users during the period from December 2017 to February 2018 were recovered. A review of the acquired data set is shown in Table 1.

4.1 Performance analysis

The performance of the proposed method is examined by performing statistical testing for sensitivity, accuracy and specificity. The statistical metrics of sensitivity, accuracy and specificity can be communicated concerning true-positive (TP), false-positive (FP), false-negative (FN) and true-negative (TN) values. Once the regulated ML models are trained, their effectiveness is assessed. The significance of each marker is graphically featured by the network in Table 2, where each section implies the events in the expected class, while each line speaks to the events in the real class:

to gauge the use of each lead to the records in the benchmark data set, thought about the going with standard appraisal measurements:

- *True positive (TP)* the number of fake users perceived by the rule as fake users
- *True negative (TN)* the number of real users perceived by the rule as real users
- *False positive (FP)* the number of real users perceived by the rule as fake users
- *False negative (FN)* the number of fake users perceived by the rule as real users

The following accuracy, false positive, precision, recall and Matthews correlation coefficient metrics are used to determine the model's efficiency.

(a) Accuracy
Accuracy is a measure of the accounts that are correctly classified:

$$\frac{(\mathrm{TN}+\mathrm{TP})}{(\mathrm{TN}+\mathrm{FP}+\mathrm{FN}+\mathrm{TP})} \tag{17}$$

(b) Precision
Precision is calculated as the number of correct positive predictions divided by the total number of positive predictions. The precision calculation is given as

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{18}$$

(c) Sensitivity
Sensitivity is calculated as the number of correct positive predictions divided by the total number of positives. The sensitivity measure is mathematically expressed as

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{19}$$

(d) Specificity
Specificity is calculated as the number of correct negative predictions divided by the total number of negatives. The specificity calculation is given as

$$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} \tag{20}$$

(e) Matthews correlation coefficient (MCC)
The estimator of the correlation between the predicted class and the real class of the samples is characterized as:

Table 2 The confusion matrix

Actual class	Predicted class	
	Real user	Fake user
Real user	TN	FP
Fake user	FN	TP

Table 3 Accuracy of the proposed EGSLA on negative, positive and neutral labels (for 800 users)

Train sentiment	Test sentiment	Accuracy
Positive	Positive	90.3
	Negative	84.1
Negative	Positive	83.4
	Negative	85.9
Neutral	Neutral	87.2

Table 4 Precision of the proposed EGSLA on negative, positive and neutral labels (for 800 users)

Train sentiment	Test sentiment	Precision
Positive	Positive	92.3
	Negative	82.6
Negative	Positive	84.5
	Negative	88.6
Neutral	Neutral	88.9

$$\frac{\mathrm{TP}\cdot\mathrm{TN}-\mathrm{FP}\cdot\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FN})(\mathrm{TP}+\mathrm{FP})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \tag{21}$$

Each of the above measures gets a substitute piece of the forecast nature of the examples that have a place with the huge class. Precision measures what quantities of tests are adequately recognized in both of the classes; by and by, it does not express if the noteworthy class is better seen over the other one. Also, there are conditions where some prediction models perform better than others, despite having lower accuracy. MCC attempts to pass on in one single value the nature of a gauge, uniting alternate measurements. Additionally, MCC uses all four quadrants of the confusion matrix.

The experiment was conducted for every rule considering two classes of accounts, the fake users and the real users. To epitomize the outcome of every experiment, some were considered as assessment metrics in light of two standard indicators, accuracy and MCC. The estimation results are shown in Tables 3 and 7, and the estimation results for the precision, specificity and sensitivity are shown in Tables 4, 5 and 6.

The accuracy of the proposed EGSLA based on positive, negative and neutral labels in train sentiment and test sentiment is given in Table 3. The neutral value is 87.2 in both the train and test sentiment. In the positive label, the positive value is 90.3 and the negative value is 84.1. Figure 3 shows the proposed EGSLA accuracy on negative, positive and neutral labels.

Table 4 shows the precision of the proposed EGSLA on negative, positive, and neutral labels for 800 users. The precision is 92.3, which is obtained in the testing sentiment of the positive level. The graphical representation of the precision value of the proposed EGSLA is shown in Fig. 4

Table 5 Specificity of the proposed EGSLA on negative, positive and neutral labels (for 800 users)

Train sentiment	Test sentiment	Specificity
Positive	Positive	89.88
	Negative	82.1
Negative	Positive	79.20
	Negative	83.4
Neutral	Neutral	87

Table 6 Sensitivity of the proposed EGSLA on negative, positive and neutral labels (for 800 users)

Train sentiment	Test sentiment	Sensitivity
Positive	Positive	90.8
	Negative	84.3
Negative	Positive	83.7
	Negative	87.9
Neutral	Neutral	88

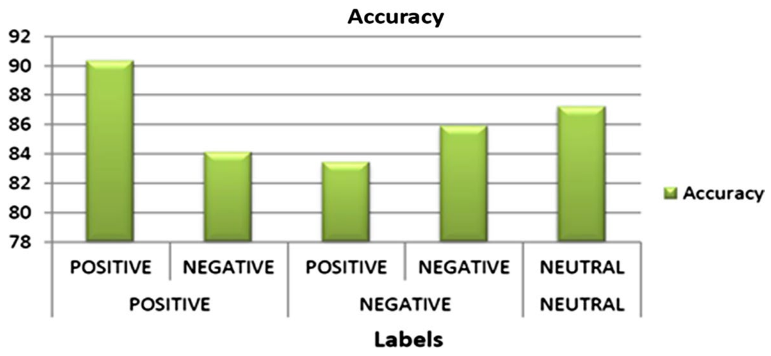

Fig. 3 Accuracy of proposed EGSLA on negative, positive and neutral labels

Table 5 shows the specificity of the proposed EGSLA on negative, positive and neutral labels for 800 users. The performance is shown in train sentiment and test sentiment phases. A pictorial representation of the specificity value is shown in Fig. 5.

Table 6 shows the sensitivity of the proposed EGSLA on negative, positive, and neutral labels for 800 users. The sensitivity value is 90.8 in positive phase. In the train sentiment positive label, the positive value is 90.8 and the negative value is 84.3. In the train sentiment negative label, the negative value is 87.9. The sensitivity value is represented as shown in Fig. 6.

Table 7 shows the MCC performance of the proposed EGSLA on negative, positive and neutral labels. In the train sentiment and test sentiment, the neutral value is 85.5. From this observation, the positive value is better than the negative value. Figure 7 shows MCC of the proposed EGSLA on negative, positive and neutral labels.

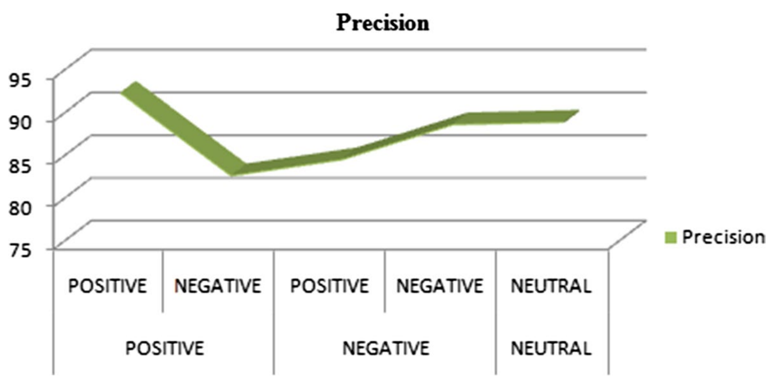

Fig. 4 Precision of proposed EGSLA on negative, positive and neutral labels

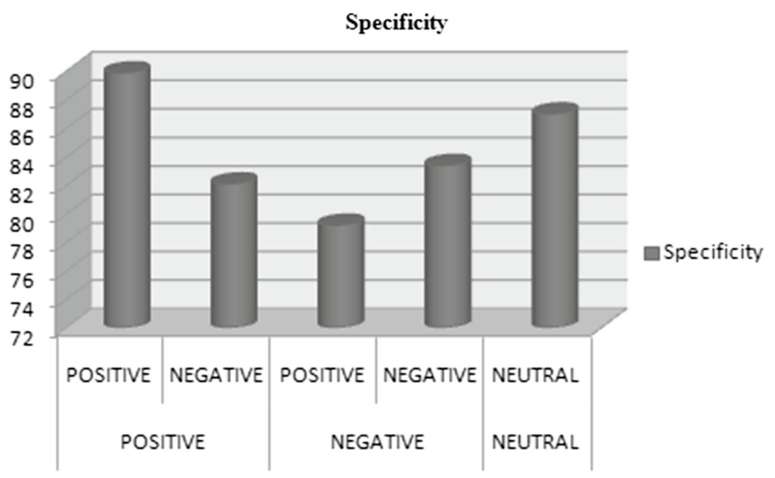

Fig. 5 Specificity of proposed EGSLA on negative, positive and neutral labels

The tweets are organized into real user and fake user groups, and based on these two groups, the tweets are labeled for examining the transition sequences. The tweets are characterized into five labels utilizing EGSLA. A group of 800 users is considered for assessment of #budget2018. The average values of each label for 800 users is shown in Table 8

The labels are given in increasing request of the total of the estimations of each measurement of the component vector of the names' centroid. Label 1 is the group whose total measurement of the centroid's element vector is the lowest. Label 5 is the group whose total of the measurement of the centroid's element vector is to the greatest.

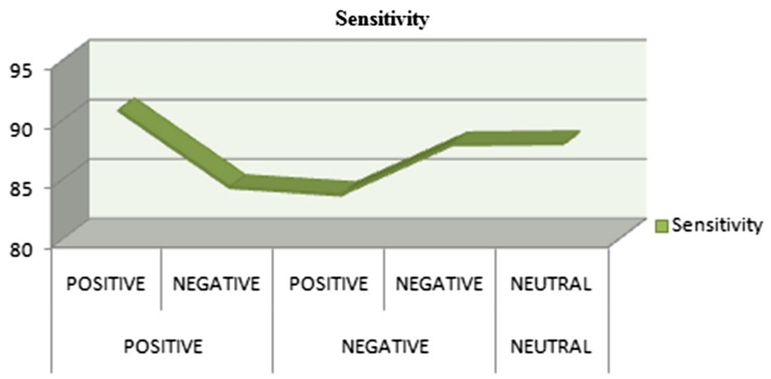

Fig. 6 Sensitivity of proposed EGSLA on negative, positive and neutral labels

Table 7 MCC measure of proposed EGSLA on negative, positive and neutral labels (for 800 users)

Train sentiment	Test sentiment	MCC
Positive	Positive	89.4
	Negative	83.2
Negative	Positive	81.8
	Negative	82.7
Neutral	Neutral	85.5

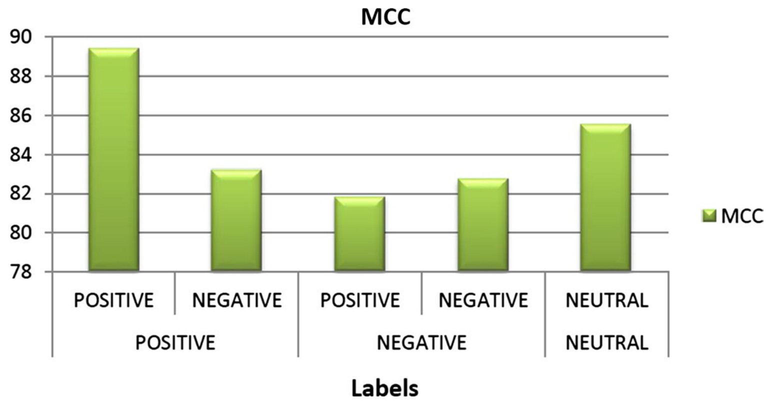

Fig. 7 MCC of proposed EGSLA on negative, positive and neutral labels

4.2 Comparative analysis

To comprehensively assess the EGSLA, we compare the detection results with those acquired by other state-of-the art fake user recognition techniques: game theory, *k*-nearest neighbor (KNN), support vector machine (SVM) and decision tree. Indeed, the EGSLA detection method outperforms the other methods for all metrics,

Table 8 Average values of each label (800 users)

Label	Post	Reply	RT	Users
1	6.253	0.138	0.130	346
2	18.238	0.309	0.106	211
3	63.917	0.409	0.080	124
4	65.471	0.318	0.144	93
5	198.817	0.371	0.117	26

Table 9 Accuracy in detecting fake users

Percentage of labeled data (%)	EGSLA	Game theory	KNN ($k=10$)	SVM	Decision tree
8	63.5	58.2	43.1	41.2	42.1
10	70.9	67.1	44.2	43.0	43.2
15	78.5	72.0	57.2	54.1	54.8
20	84.4	74.9	61.1	62.1	61.2
30	90.3	75.1	63.0	63.2	63.1

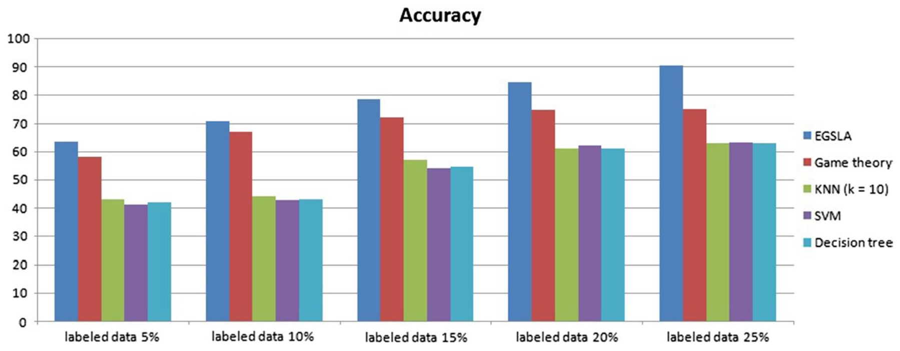

Fig. 8 Comparison of accuracy of fake user detection algorithms

achieving 89.4% MCC, 88.2% true F-measure and 89.7% false F-measure. Table 9 shows the performance comparison between different ML algorithms used to detect the fake users on OSNs.

Figure 8 shows the accuracy of various ML algorithms compared with the EGSLA algorithm.

In this experiment, it was found that if the percentage of labeled data of the anticipated characteristic is very large, the prediction accuracy can be increased by gradually increasing the percentage of labeled data. If the proclivity estimation of a property is greater than 10, users having a similar incentive on this characteristic push toward getting to be friends more successfully. The time and retweet likeness were found to be the best comparability measure expected for inferring this quality.

Therefore, increasing the weight of retweet and time closeness can increase prediction accuracy.

5 Conclusion

Semi-supervised learning algorithms demonstrate their advantages in various real-life applications because of their capacity to develop patterns with great generalizability from an extremely limited labeled sample set by making use of the unlabeled ones followed by the labeled ones for training purpose. The fake user issue is approached utilizing EGSLA. Huge quantities of feature sets are extracted by analyzing the tweets of the users. For decision making, the extracted features are classified and labeled. The results demonstrate that the proposed EGSLA algorithm distinguishes the fake user in the pool of Twitter users. The EGSLA technique was compared with other ML techniques, including game theory, KNN, SVM, and decision tree methods, to assess the performance of the proposed method, which showed that 90.3% accuracy, 88.2% true F-measure and 89.7% false F-measure were achieved using the EGSLA. This compares with accuracy of 75.1%, 63.0%, 63.2% and 63.1% for game theory, KNN, SVM and decision tree.

In the future, the proposed method can be enhanced by the analysis of user behavior patterns, and a hybrid classifier can be used in order to further improve the accuracy of fake account detection.

References

1. Hanna R, Rohm A, Crittenden VL (2011) We're all connected: the power of the social media ecosystem. Bus Horiz 54(3):265–273
2. Doan A, Ramakrishnan R, Halevy AY (2011) Crowdsourcing systems on the World-Wide Web. Commun ACM 54(4):86–96
3. Ding Y, Yan S, Zhang Y, Dai W, Dong L (2016) Predicting the attributes of social network users using a graph-based machine learning method. Comput Commun 73:3–11
4. Krombholz K, Merkl D, Weippl E (2012) Fake identities in social media: a case study on the sustainability of the Facebook business model. J Serv Sci Res 4(2):175–212
5. Chu Z, Gianvecchio S, Wang H, Jajodia S (2012) Detecting automation of Twitter accounts: are you a human, bot, or cyborg? IEEE Trans Dependable Secure Comput 9(6):811–824
6. Gilani Z, Farahbakhsh R, Tyson G, Wang L, Crowcroft J (2017) Of bots and humans (on Twitter), In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, pp 349–354. ACM
7. Qiang F, Feng B, Guo D, Li Q (2018) Combating the evolving spammers in online social networks. Comput Secur 72:60–73
8. Yang C, Harkreader R, Guofei G (2013) Empirical evaluation and new design for fighting evolving Twitter spammers. IEEE Trans Inf Forensics Secur 8(8):1280–1293
9. Aggarwal A, Rajadesingan A, Kumaraguru P (2012) PhishAri: Automatic Realtime Phishing Detection on Twitter, eCrime Researchers Summit (eCrime), 2012, pp 1–12, IEEE
10. Yan G (2013) Peri-Watchdog: hunting for hidden botnets in the periphery of online social networks. Comput Netw 57(2):540–555
11. Drevs Y, Svodtsev A (2016) Formalization of criteria for social bots detection systems. Procedia-Soc Behav Sci 236:9–13
12. Farasat A, Nikolaev A, Srihari NS, Blair RH (2015) Probabilistic graphical models in modern social network analysis. Soc Netw Anal Min 5(1):5–62

13. Ramalingam D, Chinnaiah V (2017) Fake profile detection techniques in large-scale online social networks: a comprehensive review. Comput Electr Eng 65:165–177
14. Boshmaf Y, Logothetis D, Siganos G, Lería J, Lorenzo J, Ripeanu M, Beznosov K, Halawa H (2016) Íntegro: leveraging victim prediction for robust fake account detection in large scale OSNs. Comput Secur 61:142–168
15. Escalante HJ, Villatoro-Tello E, Garza SE, López-Monroy AP, Montes-y-Gómez M, Villaseñor-Pineda L (2017) Early detection of deception and aggressiveness using profile-based representations. Expert Syst Appl 89:99–111
16. Tsikerdekis M (2017) Real-time identity deception detection techniques for social media: optimizations and challenges. IEEE Internet Comput 99:1–11
17. Kuruvilla AM, Varghese S (2015) A detection system to counter identity deception in social media applications, In: International Conference Circuit, Power and Computing Technologies (ICCPCT), 2015, pp 1–5, IEEE
18. Gera T, Singh J (2015) A parameterized approach to deal with sock puppets, In: 2015 Third International Conference Computer, Communication, Control and Information Technology (C3IT), pp 1–6, IEEE
19. Jiang X, Li Q, Ma Z, Dong M, Wu J, Guo D (2018) QuickSquad: a new single-machine graph computing framework for detecting fake accounts in large-scale social networks. Peer-to-Peer Netw Appl 1–18
20. Yuan W, Yang M, Li H, Wang C, Wang B (2018) End-to-end learning for high-precision lane keeping via multi-state model. CAAI Trans Intell Technol 3:185–190
21. Shi Q, Lam HK, Xiao B, Tsai SH (2018) Adaptive PID controller based on Q-learning algorithm. CAAI Trans Intell Technol 3(4):235–244
22. Wang K, Zhu N, Cheng Y, Li R, Zhou T, Long X (2018) Fast feature matching based on r-nearest k-means searching. CAAI Trans Intell Technol 3(4):198–207
23. BalaAnand M, Karthikeyan N, Karthik S (2018) Designing a framework for communal software: based on the assessment using relation modelling. Int J Parallel Prog. https://doi.org/10.1007/s10766-018-0598-2
24. Solomon Z, Sivaparthipan CB, Punitha P, BalaAnand M, Karthikeyan N (2018) Certain investigation on power preservation in sensor networks. In: 2018 International Conference on Soft-Computing and Network Security (ICSNS), Coimbatore, India, https://doi.org/10.1109/icsns.2018.8573688
25. Sivaparthipan CB, Karthikeyan N, Karthik S (2018) Designing statistical assessment healthcare information system for diabetics analysis using big data. Multimed Tools Appl

Affiliations

M. BalaAnand[1] · N. Karthikeyan[2] · S. Karthik[3] · R. Varatharajan[4] · Gunasekaran Manogaran[5] · C. B. Sivaparthipan[3]

N. Karthikeyan
profkarthikeyann@gmail.com

S. Karthik
profskarthik@gmail.com

R. Varatharajan
varathu21@yahoo.com

Gunasekaran Manogaran
gmanogaran@ucdavis.edu

C. B. Sivaparthipan
sivaparthipanece@gmail.com

[1] Department of Computer Science & Engineering, V.R.S. College of Engineering & Technology, Viluppuram, India

[2] Department of MCA, SNS College of Engineering, Coimbatore, India

[3] Department of Computer Science & Engineering, SNS College of Technology, Coimbatore, India

[4] Department of Computer Science & Engineering, Anna University, Chennai, India

[5] Department of Computer Science, University of California, Davis, USA