# TITLE

AUTHOR

DATE

## Introduction

The ubiquity of social media makes it easy for people to communicate with each other and stay informed about global events. Many platforms rely on crowdsourced information for promoting a business' services and/or products. However, this provides a vulnerability that commercial entities can easily exploit for their own interests. For example, a company could use a social media bot to promote its own products and write glowing reviews on websites such as Yelp and Google Reviews while posting negative reviews on its competitors' products. This is a lose-lose situation for both businesses and consumers; businesses must constantly defend themselves from the onslaught of negative feedback while consumers will lose trust in information from social media. Therefore, in this survey, we will discuss methods that can be used to differentiate between real users and non-human users.

At the general level, this is a problem of *anomaly detection*—finding users that are unlike the rest. Historically, there have been many approaches to anomaly detection, such as Poisson, SFP, Pearson's $\chi^2$, and CNPP. We will analyze the pros and cons of these existing methods. Next, we will introduce state-of-the-art models that attempt to address some of the shortcomings of these more mature models and highlight their relative improvements. These current methods fall under a few major categories, including temporal, Bayesian, and group.

## Background

Each type of anomaly detection method introduces its own set of definitions and fundamental ideas. Temporal-based approaches primarily use the **inter-arrival times** (IAT) between postings to identify patterns that could be used for detecting anomalies. The IAT distributions between humans and bots are compared to classify users.

To fully understand group-based approaches, a **group** must be defined. A group is a mixture model of a user behavior mixture model. The **role mixture rate** is an inference of group membership and role identity of each individual inside of a group. A **group anomaly** occurs when the role mixture rate is significantly different than a normal group role mixture rate. Rather than identifying individual anomalies, a group-based method classifies all individuals belonging to a group as suspicious if an anomaly is detected.

There are a few other different types of anomalies that a user can be classified as. An individual is considered to be a **point anomaly** if it is anomalous with respect to the rest of the data. If an individual is anomalous in a specific context, but not otherwise, it is defined as a **contextual anomaly** or **conditional anomaly**. A **collective anomaly** are a group of individuals that may not be considered to be anomalies independently, but the occurrence of them together is considered to be an anomaly.

The optimal type of anomaly detection algorithm to use is dependent on the input features that are available. For example, if given continuous or sequential time-series data, a temporal

approach may have the best performance. On the other hand, if only given categorical data, machine learning techniques such as KNN or logistic regression might be better. Input features are spatial if each data instance is related to its neighboring instance(s); for temporal-based data, these features are defined as *spatio-temporal* data. A combination of different types of input features could be solved using a Bayesian approach. Entities that have quantifiable relationships between them are probably best solved by a graph-based approach that uses either node-attributed or edge-attributed values to detect anomalies. Lastly, if classification labels are unavailable, an unsupervised method may yield better performance.

While anomaly detection seems to be intuitively simple from a high-level perspective, there are many factors that make it quite difficult. To classify something as an anomaly, there needs to be concrete definitions of what is considered "normal" and what is considered an "outlier"; these definitions can vary significantly depending on the underlying context of the problem. In the context of social media, a "normal" user is defined to be a human and an "outlier" is a bot. Although these are clearly defined contrasts, there are many bots that attempt to behave as humans, camouflaging them from many anomaly detection techniques.

Furthermore, the behavior of human users potentially varies on different social media platforms. For example, posting times of the same user on Facebook and Reddit could be very different because a user's Facebook audience is generally people that he/she knows in real life, while Reddit has an increased sense of anonymity. Because the features of a "normal" user varies platform-to-platform, an anomaly detector will likely have to be retrained and/or reconfigured for each individual case.

Another difficulty is that human users are likely to have distinctive posting patterns. The noise caused by natural variation of human personality may make it difficult for an anomaly detector to draw the line between a "normal" user and an "outlier". For example, one legitimate user could constantly be posting on social media, while another may only post something once every few weeks. While the typical bot behaves more like the first type of user, a bot creator could easily manufacture the behavior of a user that posts infrequently by creating multiple bots. An anomaly detector would probably be able to detect the first bot (and might even trigger a false positive for the first type of user), but would unable to detect the "array" of bots–although they end up having a similar aggregate post frequency.

The evaluation of each anomaly detection method will be similar. The most important performance metric is anomaly detection accuracy; if a method is unable to correctly differentiate between humans and bots, then it is not very useful. The predictions from each method must be considered as well; because there is potentially a fine line between the features a human and a bot, a simple binary classification (human, bot) may not be as informative as a "score" for how anomalous a user/group is. Because real-world datasets have millions of data points that must be accounted for and processed, scalability in terms of both space and time is very important. Methods should be robust, as noise is quite common in real-world data, among both "normal" users and "outliers". Furthermore, some methods have the assumption that observations are i.i.d., which will be considered a weakness.

## Models, Approaches, Algorithms

Methods developed before 2014 will be introduced here as both a basis of improvement and reference for state-of-the-art methods.

### Cascading Non-homogeneous Poisson Process (CNPP) – 2009

CNPP is a variation of the double chain hidden Markov model and is used for understanding the behavior of individuals and detecting outliers. Specifically, this model is used to detect

variability among individuals by using data from email communication. First, an inference algorithm is used to estimate model parameters. This model is then applied to the email data. From this experiment, the authors found that individuals can be classified into specific types and that users can be clustered together based on their model parameters which leads to easier ways to detect outliers.

## Power-Law – 2005

The power-law distribution is a non-Poisson distribution that models human behavior, which consisted of short bursts of activity separated by long periods of inactivity. The explanation for this type of distribution is the fact that humans execute tasks by their own set of priorities. Thus, tasks wait an uneven amount of time before execution.

## Self-Feeding Process (SFP) – 2013

### Approach

The SFP model combines the benefits of the non-homogeneous Poisson Process (PP) from CNPP and the power-law distribution. This is because the non-homogeneous PP works well to model user communication where short-term communication is PP, but there are long periods of inactivity where the PP rate should change. The power-law distribution captures a user's long periods of inactivity which are seen as a heavy tail in the IAT probability density function.

### Model

It attempts to accomplish four goals:

1. Realism of marginals
2. Realism of the local Poisson distribution
3. Avoiding the independent and identical distribution (i.i.d.) fallacy
4. Requirement of only a few parameters

To achieve the first goal and to better view the dataset without losing information, SFP computes the odds ratio (OR) as such:

$$OR(t) = \frac{CDF(t)}{1 - CDF(t)}$$

where the set of IATs are used to compute the OR for each percentile of the data.

For the second goal, the Poisson distribution is built into the SFP model. The third goal is obtained by building a Markov process where each even of activity influences the next event.

The Poisson process model is defined as the following:

$$\Delta_i \sim exp(1/\lambda)\ for\ i = 1 \ldots n\ events$$

SFP's piecewise Poisson model updates it as:

$$\Delta_1 \sim \mu$$

$$\Delta_i \sim exp(\Delta_{i-1} + \mu/e)$$

where e is the Euler constant.

As seen from the model, the last goal is achieved, as only the mean of the data set is required as input. The model can be generalized for other applications that only need two parameters, $\mu$ and the OR slope $\rho$:

$$\delta_1 \sim \mu$$

$$\delta_1 \sim exp(\delta_{i-1} + \mu^\rho / e)$$

$$\Delta_k \sim \delta_i^{1/\rho}$$

## Pearson's $\chi^2$ – 2011

### Approach

This paper creates a model to determine whether a Twitter account uses automation to publish tweets for spamming purposes. The authors come to the result that 16% of active accounts show automated behavior. A bot is defined as a computer program that automatically publishes a significant portion of their tweets. They believe that automated accounts have a specific timing pattern that differs from real users. For example, real users might tend to post uniformly throughout whereas automated accounts might post at the beginning or specific interval of the hour. Automated accounts tend to have non-uniform timing or excessive uniformity.

### Model

The model used to detect whether the posted times match with those of real user is Pearson's $\chi^2$ test. The test returns a p-value; if this value is below a certain threshold, the account belongs to a real user and if this value is higher than this threshold, then this account is most likely automated. For this test, the threshold was chosen to be 0.1% specifically to avoid false positives and detect accounts that post with excessive uniformity. A false positive occurs when the test says an account is automated when it is actually organic. However, keeping the threshold at 0.1% has its drawbacks as it will be difficult to classify hybrid accounts which contain both manual and automated postings. A false negative tends to be higher because of these hybrid accounts being classified as organic accounts instead of automated accounts. Another drawback of this test is that an automated account can pass this test if they post uniformly throughout the hour and second.

All accounts can display some form of automation depending on temporal factors such as time of day. In order to accurately determine automated accounts, one will need to examine tweets spanning over a couple of weeks or months, but the authors have left this to do in the future. A previous study had a model where accounts that published more that 150 tweets per day were automated. After Twitter started using this model, the number of automated accounts dropped down to 14%. When looking at only Verified accounts, only 6.9% were determined to be automated, typically due to tweeting news or reminders throughout the day. For those in Trending Topics, only 4.7% of the accounts were automated, which could be attributed to the fact that Twitter prevents automated tweets from appearing in trending topics. In the future, the authors wanted to investigate the frequency of words and compare them between automated accounts and organic accounts.

In conclusion, Pearson's $\chi^2$ test can detect automated accounts which have distinct timing patterns, which found that 16% of public accounts are automated. The authors suggest to use this model to prevent spam and other use.

## Human, Bot, or Cyborg? LDA Approach – 2010

### Approach

The paper is an attempt to tackle the problem of determining whether a given Twitter user is a human, bot, or cyborg, defined as follows. A human user is a Twitter account whose tweets are completely posted by a human. A bot is an account whose posts are completely automated. A cyborg is an account that may have both automated activity and human activity. The researchers trained their model using 1,000 accounts for each human, bot, and cyborg, amassing over 8 million tweets, and used a sample of 500,000 users, with upwards of 40 million tweets, to then estimate the composition of Twitter at the time. Ground truth for the training set was determined manually.

### Model

The researchers take a tiered approach, splitting the problem into four steps:

1. Use entropy to determine the complexity of user behavior
2. Apply a machine learning component to detect spam-like features
3. Use various account properties to detect outlying users
4. Combine features generated from the previous three steps to come up with a model for humans, a model for bots, and a model for cyborgs.

By applying the data on each of these models, resulting in a human-score, bot-score, and cyborg-score, they can determine which category a user falls in.

The entropy rate of a Twitter user refers to the complexity of the distribution of a user's tweets over time. A low entropy rate indicates that the tweets are posted fairly regularly, whereas a higher entropy rate implies a more random process. The researchers approximate the entropy rate using empirically-derived probabilities.

Spam score is the classification of a user's tweets as containing overly "spammy" language. The researchers use a Bayesian classifier to classify the text, assuming the presence of words are conditionally independent of each other. This model empirically calculates the probability that a tweet is spam given the words in it using the package CRM114, which includes an OSB classifier, which is similar to a traditional naive Bayes classifier for text, but classifies based on pairs of words.

There are a number of other features examined and used in this model. The first is the URL ratio, i.e. the ratio of how often a user includes an external URL. The researchers also examine the percentage of tweets posted using manual devices, i.e. web and mobile devices instead of via the Twitter API. Another highly discriminatory features is the automated post percentage. Rather than taking into account what devices a user uses, this feature is the percentage of posts to a user's Twitter account that follow from the RSS feed of another site (e.g. the user's blog) via services like Twitterfeed or Twitter for Wordpress. The last feature they use is the followers to friends ratio. The intuition is that bots add many users as friends, but few users follow a bot back, so bots will often try to follow as many users as possible. However, Twitter has imposed limits on the ratio of followers over friends to suppress these bots, so often, a bot will unfollow users who do not follow them back, keeping their ratio suspiciously close to 1:1. They find that bots rarely have more followers than they have friends.

By feeding these features into a decision maker, the researchers come up with a model, based on linear discriminant analysis and forward stepwise analysis. It is stepwise, as the model is built step-by-step. At each step, all features are evaluated, and only the one that contributes the most to discrimination is added to the model. Linear discriminant analysis takes labeled data

and assigns weights to each feature that discriminates a single class from the others. A linear combination of the features based on these weights yields a score determining how human-like, bot-like, or cyborg-like a user account is. By taking the highest of these scores, the model predicts a label for these features.

#### Results

The researchers report high accuracies, predicting true humans, bots, and cyborgs 94.9%, 93.7%, and 83.8% of the time respectively. Their predictor can perfectly distinguish bots from humans, but performs worse in differentiating humans from cyborgs, or bots from cyborgs. Applying this model to their 500,000 users, they estimate the ratio of human, cyborg, and bot on Twitter to be 5:4:1.

The authors outline the pitfalls with their model that led to misclassifications. Spam score is a tricky feature to use, since even human users use "spammy" words in advertising their own content. Additionally, bots and cyborgs can use very naturally scripted language, tricking their spam classifier. Additionally, the entropy rates of humans could be low if a user often posts at certain times.

Because the LDA approach uses features that require more effort to impersonate, it is not trivialized by simple workarounds such as adjusting the tweet times. However, it is a basic decision maker because it has a linear model of features. Also, any of the features could yield false positives or negatives; for example, a human user that only posts via a blog RSS feed could be classified as suspicious. On the other hand, some of the features were ineffective (i.e. low coefficients for Bayesian text, followers to friends ratio, URL ratio) Additionally, the relationship between humans, bots, and cyborgs is not considered, so the model only infers it from the LDA weights.

## Current State-of-the-art

While state-of-the-art methods use the same basic ideas and principles of older models and fall under the same set of categories, they attempt to improve on them in some way.

### Rest-Sleep-and-Comment (RSC) – 2015

#### Approach

This paper investigates the differences between humans and bots based exclusively on the timing of posts from two specific social media services: Reddit and Twitter. IATs are used for measuring temporal data. Using the distribution of IATs, four patterns were identified:

1. Positive correlation between consecutive IATs
2. Heavily-tailed distribution
3. Periodic spikes
4. Bimodal distribution

The patterns were common among various social media services and were identified through analyzing IAT distributions from timestamp data. A model was then designed to generate synthetic timestamps whose IAT fits the data distribution and matches all patterns of actual human users. Non-human users could then be detected using this model.

**Model**

RSC is a generative model that matches these patterns to classify anomalies based only on temporal activities.

First, a sequence of synthetic IAT must be generated. This is done through the Self-Correlated Process (SCorr), which is a stochastic process that generates consecutive IATs that are correlated, which is different than sequences generated from a Poisson Process or a priority queue based model. This is meant to match the positive correlation pattern from actual human data. In SCorr, the duration $\delta_i$ between two events is sampled from an exponential distribution with rate $\lambda$ depending on the previous IAT $\delta_{i-1}$. It uses the correlation parameter $\rho$ to control the dependency between consecutive IATs. When $\rho \to 0$, SCorr reduces to a Poisson Process with rate $\lambda$ and when $\rho = 1$, it reduces to a Self-Feeding Process.

Next, for modeling the bimodal IAT distribution and period spikes from actual data, three states were defined for the RSC algorithm: active, rest, and sleep. While RSC is in the active state, it generates posting events with a probability $p_{post}$ or null events with probability $1-p_{post}$ at every time interval $\delta_i^A$. The intervals $\delta_i^A$ and $\delta_{i+1}^A$ are correlated and generated using SCorr. While RSC is in the rest state, null events are generated at every time interval $\delta^R$, which contributes for incrementing the IATs between postings. While RSC is in the sleep state, a single null event is generated after a time interval $\delta^S$, which corresponds to the time required to advance the clock time $t_{clock}$ until the next wakeup time $t_{wake}$.

Because RSC is based on a 24-hour cycle, $t_{clock}$ spans between $0:00h$ and $23:59h$. Under the assumption that $t_{wake} = 0$, the parameters $t_{wake}$ and $t_{sleep}$ can be replaced by a single parameter $f_{sleep}$, which corresponds to the fraction of the day that RSC considers as sleep-time. Using this parameter, $t_{sleep} = f_{sleep} * 24h$ when $t_{wake} = 0$. When a user is active, there is a probability $p_r$ that it will transition into the rest state and $1 - p_r$ probability for it to remain active. While in the rest state, there is a $p_a$ probability to transition into the active state and $1-p_a$ probability to remain in the rest state. However, a user will always transition from rest to sleep when $t_{wake} > t_{clock}$ and $t_{sleep} < t_{clock}$, meaning that the current clock time falls within the sleep time. After the sleep state ends, the user will always transition into the rest state.

**Algorithm**

To estimate the parameters for RSC, an algorithm based on fitting the histogram of the observed data IAT is used. First, synthetic timestamps are generated using the RSC model. Next, a log-binned histogram of IAT is computed for real and synthetic data. The width $w_i$ of the i-th bin is wider than the previous bin by a fixed factor $k$, meaning that $w_i = w_{i-1} * k$. The counts of IAT in each i-th bin are denoted as $c_i$ and $\hat{c}_i$ for real and synthetic data respectively. With the Levenberg-Marquardt algorithm, the parameter values $\theta$ that minimize the square difference between the synthetic and real data bin counts are calculated: $min_\theta \sum_i (c_i - \hat{c}_i(\theta))^2$. Using logarithm binning when approximating the PDF allows the parameter estimation method to match the heavy tail pattern and daily spikes found in real data.

Next, RSC-Spotter uses RSC to detect bots. It compares the distribution of IAT of each user to the aggregated distribution of IAT of all users in the dataset. Users that have an IAT distribution significantly different from the aggregate are flagged as anomalies and potential bots. It first estimates RSC parameters using the aggregate IAT data and then generates $n_i$ timestamps, where $n_i$ is exactly the number of timestamps for user $U_i$. Then, RSC-Spotter calculates the dissimilarity between the synthetic timestamp distribution with the users' actual timestamps.

Finally, it has to be determined whether or not the dissimilarity value $D_i$ classifies that user $U_i$ is a human or a bot. Given a training set of users labeled either as bots (positive) or humans

(negative), a Naive Bayes classifier is trained to estimate the posterior probability $p_{bot}$ that a user is a bot, using the dissimilarity values $D_i$ as features for the classifier. If $p_{bot}$ is greater than a decision threshold $p_{thresh}$, then the user is classified as a bot. The threshold $p_{thresh}$ is estimated by assigning a cost $c_{FN}$ to false negative errors and a cost $c_{FP}$ to false positive errors. A false positive is when a human is classified as a bot and a false negative is when a bot is classified as human.

In conclusion, RSC was quite successful in differentiating between humans and bots. With only temporal activity, RSC was able to detect non-human users with 96.5% precision on Reddit and 94.7% precision on Twitter.

## BIRDNEST – 2015

### Approach

Because IATs do not completely account for bot behavior, the authors proposed BIRDNEST as a Bayesian-based method to approach the anomaly detection problem. BIRDNEST expands on the pure temporal-based approach by using both IAT and rating distributions.

When determining the suspiciousness of a user, one may first ask what the rating distribution or IAT distribution is for a given user. For example, a user that rates all ones or all fives would have a single peaked rating distribution and a user that rates all fives for its own company's products and all ones for its competitors' products would have a bimodal rating distribution. BIRDNEST approaches user rating-fraud detection by solving these questions with Bayesian inference. The user's rating distribution is the prior distribution and the belief of the user's true long-term behavior trends is the posterior distribution. The goal is to detect a typical bot's behavior of a narrow posterior distribution towards one extreme of the rating range or the other since this behavior is vastly different from a typical normal distribution or even a newbie legitimate user.

### Model

The BIRDNEST model has two phases: fitting the Bayesian model (BIRD) to a given data set and creating the metric of suspiciousness called the Normalized Expected Surprise Total (NEST). It expects as input data the following: a matrix of users' ratings (rating from 1 to n) where the user is indexed from 1 to m and the corresponding matrix of timestamp of the rating. The timestamp data is pre-processed to calculate the IATs of each user.

The BIRD generative model is a mixture model where there are $K$ clusters. Each of these clusters are trained by the user training data to represent various normal user prior distributions using the Dirichlet prior distribution. Once the mixture model converges, the posteriors are calculated based on the Dirichlet distribution property.

The NEST part of the algorithm uses the BIRD mixture model posterior distributions to calculate the suspiciousness of a user. The metric surprise is defined as the negative log likelihood the user's distribution would be generated by the global mixture model distribution. Because rating and time have different ranges, the surprise metric of rating and time are normalized and summed for the final NEST value. The smaller the NEST value, the less suspicious the user is and it is less likely that the user is a bot.

### Algorithm

The algorithm first maximizes the overall likelihood function to calculate the best posterior marginals. Then, the NEST metric is calculated to quantitatively determine how certain the model is of the user being fradulent.

1. Determine the number of clusters, $K$, by the Bayesian Information Criterion (BIC).
2. Optimize the mixture model for the given data set by iteratively adjusting the hyperparameters and recalculating the mixture weights until convergence
3. Calculate the posterior distributions of $p_i$ and $q_i$ using the conjugate prior property of Dirichlet distributions.
4. Calculate the surprise for rating and IAT distribution, normalize distributions by the standard deviation, and sum to create the NEST metric of suspiciousness.

While BIRDNEST is able to compute the degree of suspiciousness of a user rather than simply classifying it as a human or a bot, it assumes that a user's rating events are independent of each other (i.i.d. fallacy).

## Group Latent Anomaly Detection (GLAD) – 2014

### Approach

In this paper, the authors' goal is to determine group anomalies by proposing the GLAD (Group Latent Anomaly Detection) model, which uses a hierarchical Bayes model. Unlike previous models, GLAD takes in pair-wise and point-wise data as input and determines the groups and group anomalies simultaneously. Until now, people were only trying to detect individual anomalies; however, detecting group anomalies early and efficiently is also important in order for preventing group attacks. There are three main observations made towards group anomaly:

1. Both point-wise data and pairwise data exist in social media
2. Group anomaly is harder to detect than individual anomaly
3. Individuals are dynamic which makes detecting group anomaly harder to find

The authors first look at previous models that were used to detect group anomaly such as Multinomial Genre Model (MGM), Latent Dirichlet Allocation (LDA), Flexible Genre Model (FGM), and support measure machine (SMM). Both MGM and LDA define a group as a mixture of Gaussian distributed topics and gives each group a certain mixture rate. In order to determine the role mixture rate for each individual, we infer their role identity and group membership. If this mixture rate is significantly different than a mixture rate of normal groups, then we can classify this group as an anomaly. All of these models first determine the groups and then discover anomalies in a two-step process, but GLAD attempts to do this process simultaneously. However, all of these models, including GLAD, are only able to handle static data, which means that there needs to be another model which can handle both dynamic and temporal data.

### Model

In general, the GLAD model first identifies the normal mixture rates and then observes the likelihood of the observations given these normal mixture rates. A particular group is an anomaly if this likelihood value is low.

Each person can belong to a group given a membership distribution which is defined by the Dirichlet prior. The multinomial distribution calculates the likelihood of that individual belonging to a specific group. Two people can be linked based on their group identities. Each group has a role mixture rate and each person can have multiple roles and multiple memberships. A limitation of GLAD is that it can only determine group anomaly if the groups stay static.

A modified version of the GLAD model, d-GLAD, can handle both dynamic data and detect mixture rates with respect to time with these groups. This model uses a variational Bayesian

method and a Monte Carlo sampling technique. Essentially, at a certain time, a GLAD model is applied and the mixture rates are stored. This is done with the multivariate Gaussian distributions. Each mixture rate at a certain time is determined from the Gaussian distribution of the previous time mixture rate. When there is a significant change to the mixture rates, then there is an anomaly.

Both GLAD and d-GLAD were tested on synthetic and real data. In a dataset of scientific publications, GLAD was able to detect anomalous papers; in a dataset containing US Senate Voting, d-GLAD was able to detect changes in party affiliation over time. However, there are certain limitations such as group anomaly is difficult to detect without labeled data and the d-GLAD model is computationally expensive and is not scalable to a large dataset.

## ND-SYNC – 2015

### Approach

The authors first survey different forms of fraud-detection on Twitter, citing two common tactics that are explored in several other papers: simple feature-based detection and collective anomaly detection. ND-SYNC is the first to be textual content-agnostic, graph structure-agnostic, user attributes-agnostic, parameter-free, unsupervised, specifically designed for RT-Fraud problem, and a well-performing detector for synchronicity fraud.

### Model

This paper investigates and proposes a method for detecting Twitter fraud as it pertains to fraudulent retweet activity (RTFraud). By analyzing numerous features captured over large retweet threads, they define and measure of the concept of synchronicity across multiple features. The authors tackle this by formulating RTFraud as a special case of the synchronicity fraud problem (SyncFraud). They extract features for each retweet thread and assign a suspiciousness score based on the similarities to other threads, assuming that organic behavior is the norm.

### Features

The *synchronicity* of a group is a measure of closeness between all of its members. The *normality* of a group with respect to a superset is a measure of closeness between members of the group and members of the superset. Finally, they define the residual score as the difference between the synchronicity and mathematically-defined lower bound on synchronicity. According to the SyncFraud problem, suspicious groups are those that exhibit highly synchronized characteristics.

To tackle the RTFraud problem specifically, the authors experimented with a large number of features and determined the seven most significant features to be:

- Number of retweets
- Response time of first retweet
- Lifespan, constrained to three weeks
- RT-Q3, the time to garner the first three quarters of the retweets
- RT-Q2, the time to garner the first half of the retweets
- Arr-MAD, mean absolute deviation of IATs
- Arr-IQR, inter-quartile range of IATs

By projecting their data onto two of these seven features, the authors came up with visualizations of their data that helped them find intuitive patterns in the data, namely lifespan vs. Arr-MAD and retweets vs. response time.

**Algorithm**

One of the major components in determining suspiciousness across all features is to do a sweep over all feature subspaces. Given p features, there are $2^p$ subspaces to examine, and the original p-dimensional entities must be projected onto each of them. The researchers show that even 3-D and 2-D subspaces are effective in building an intuition of the relationship between features.

The suspiciousness of a group with respect to a feature subspace is determined as the residual score of the projection of that group on that subspace. Intuitively, if the group is very synchronized in a given subspace, the group has a high suspiciousness score for that subspace. This is calculated for each group, for each of the $2^p$ subspaces. Unlike other algorithms, which usually detect outlying groups based on low inter-synchronicity, ND-Sync looks for high intra-synchronicity, which the researchers argue is a better indicator, as they find that that normal users are approximately at around the same level of inter-synchronicity. Essentially, real data has no "norm" for the data to fit, so detecting outliers proves fruitless. Instead, this approach looks for structure.

Given the the suspiciousness scores over all subspaces for each group, outliers are detected as groups that deviate from other groups, based on the ROPCA-AO algorithm explained below. The intuition behind this is that all groups (retweet threads) should have similar synchronicities for each feature subspace. If a majority of the groups have low synchronicities across all subspaces, a group with a high synchronicity in some subspaces would be considered suspicious.

The ROPCA-OA algorithm is defined as follows. Given a set of high-dimensional data, this method applies an SVD, yielding the principal components of the data. The strongest $k$ features are kept, and a k-subspace is computed. The algorithm calculates the orthogonal distance of each observation to its projection in this subspace, and the distance of a given observation from a bulk of the observations; if they are past a certain threshold, the point is considered an outlier.

**Results**

The researchers report between 95% and 97% accuracy based on a dataset collected of the span of six months. They found accounts that had a large number of retweet threads with a large number of retweets in each, arriving at 298 users (270 honest, 28 fraudulent). Honesty was determined by manually inspecting each account, based on spammy links and clear fabrication of identity. Over the course of six months, this yielded 134,022 retweet threads, with almost 12 million retweets total. After applying the algorithm, among the fraud accounts that are caught, a majority were near-idealized cases of bots, mostly for promotion or advertisement. The remaining accounts were more subtle, often belonging to cyborgs rather than true bots. The false positives detected by ND-Sync belonged to politicians and media accounts, who tend to have less organic behaviors.

ND-SYNC is able to detect fraud with high accuracy and was able to yield intuitive explanations by plotting features against each other. However, it was unable to differentiate between cyborgs from bots and fails with certain humans (i.e. politicians and people who advertise); it also assumed that synchronized data is always fraudulent, which is not always true. Also, the dataset used had a limited number of fraud-labeled users, which may have skewed the accuracy

upward.

## DARPA Twitterbot Challenge Top 3 SVM Approach – 2015

### Purpose

The problem of the DARPA Twitterbot Challenge was: given logs of users that posted in vaccination-related threads, determine which were "influence bots"—bots that are designed to generate conversation about a given topic. The proposed problem was not constrained to be completely automated and assumed there were bots present (although the exact number was not made known).

### Features

In order to gain insight into which approaches would potentially work, the teams propose a variety of different features to examine and compare.

Syntax features: similarity to known NLP generators, number of hashtags, user mentions, links, or special characters, number of retweets by user, geo-enabled, whether tweet ends with punctuation, hashtag, or link

Semantic features: number of user posts related to vaccination, user's average sentiment score on tweets with the word "vaccination", contradiction rank, overall positive sentiment strength, most frequent topics tweeted about, number of languages tweeted in, sentiment inconsistency (Latent Dirichlet Application for topic discovery, AVA and OASYS for sentiment)

Temporal behaviors: flipping sentiment over time, entropy of inter-arrival times, duration of longest session by a user, number of dropped followers over time

User profile features: number of posts, number of followers and followings, similarity to known bots (Jaccard similarity, cosine similarity)

Network features: deviation of sentiment from followers, number of bots followed by user

### Algorithm

In order to detect bots, the top two teams first narrowed down the data to detect a handful of bots. For this, they applied various data visualizations to determine outlying data, leading them to select fairly obvious bots.

The teams make the assumption that the bots all have similar behaviors and use various clustering algorithms to find similar accounts to known bots. USC uses KNN search to create a graph of similar users based on a number of features; it then applies modularity maximization to identify groups of accounts that has many connections within the same cluster. The Sentimetrix team uses DBSCAN to cluster around the manually selected bots. Once a certain number of humans and bots are found using the previous algorithms, all teams employed the use of a support vector machine to determine whether there were any additional bots.

Indiana's team attempted to use a multi-armed bandit approach, assigning a "bot score" to each test point, and adjusting the weights such that the accurate arms gained weight and inaccurate ones lost weight. This approach is an online approach, so it could dynamically learn based on each additional input.

### Results

All of the participating teams were able to correctly identify the bots. However, the one challenge that many teams faced was to determine when to stop guessing, as they were not

informed beforehand how many bots there are. Only one team was able to detect the bots exactly.

While the algorithms presented were able to achieve up to 100% accuracy and they took almost all types of features into account, there were some major downsides as well. Most importantly, they assumed that bots were present and because the problem was very specific, many of the approaches would only achieve high performance for this exact problem. Furthermore, some components of the algorithms used human judgment to augment the bot identification process; a semi-automated pipeline would be less efficient for real-world use cases.

## PCA – 2014

### Approach

The paper "Toward Detecting Anomalous User Behavior in Online Social Networks" uses an unsupervised learning algorithm called Principal Component Analysis (PCA). The purpose of PCA is to find the best projections of a dataset that show the most variation given a set of features. For anomaly detection, PCA is utilized by projecting where bot-data differs from normal-user behavior.

### Features

Features for this model can consist any of the following: temporal features consist of a histogram of a per-time-unit (e.g. per-day) basis, spatial features consisting of a histogram of a non-temporal feature such as categorizes of Facebook pages that were liked by a user, and spatio-temporal features which is the observation of a spatial feature over time.

For temporal features, each time-bucket is a different dimension, allowing PCA to model IATs, weekday patterns, weekend patterns, rate of change of activity, etc. as a linear combination. Spatial features are histogram buckets of observing an attribute where each bucket is a different dimension. Spatio-temporal features is a histogram where each bucket is a spatial distribution at that time. To summarize the distribution within a bucket, entropy can be used where the probability of bucket $i$ is represented as $p_i(-\sum_i(p_i) * log_2 p_i)$.

### Model

PCA is an unsupervised learning algorithm which finds the most varied projects of a given data set. This model is represented as a linear combination of the most uncorrelated features (aka principal components or singular values) where the first principal component has the maximum variation. Human behavior can be described in a small amount of dimensions (k-dimensions) which this study shows because five principal components were enough to describe 85% of the data's variance.

Anomalies are determined based on the distance from the normal subspace. The normal subspace is the projection of the first five principal components (more generally k-components). The residual subspace is thus the projection of the last principal components (more generally k+1 through n-1 components). Therefore, the residual subspace detection is determined by the distance between a given point and the normal subspace. If the distance is above a certain threshold, then the user is classified as anomalous.

### Results

PCA was able to detect bot-users with a high accuracy and compromised legitimate users with greater than 50% accuracy without a labeled data set because it was able to detect the most weighted features. However, classification varies on the choice of threshold, which may be

a difficult hyperparameter to tune based on the specific problem at hand. Also, if the user data is too variable, a bot could potentially hide in the noise.

## Detecting Lockstep Behavior – 2014

### Approach

The main idea of the approach is to leverage social graph data to identify the "lockstep behavior pattern" (groups of followers typically following the same group of followees with little to no other activity). When there is no lockstep behavior, the adjacency matrix does not have large, dense sections of the matrix; this means there are no followees with an abnormal amount of followers who steadily follow only the same subset of followees. However, if a small group of followees have a large amount of similar followers that only follow those groups, this behavior is known as non-overlapping lockstep behavior. There is the potential for some overlap lockstepping behavior where a group of followers follow a large amount of the same followees but each follower does not have the exact same set of followees as the non-overlap lockstep behavior pattern. Instead, a subset of these followers show the non-overlap behavior for some followees while other subsets show the same non-overlap behavior for some other followees.

### Model

A fundamental part of this approach is the use of spectral-subspace plots. Given an adjacency matrix that is $NxN$, spectral-subspace plots project these points from $N$ dimensions into 2 dimensions spanned by two singular vectors. For a non-lockstep behavior spectral-subspace, points cluster around the origin. However, for the non-overlap lockstep "block" behavior has a spectral-subspace plot that shows a "ray" pattern from two of its singular vectors. In addition, the overlap lockstep "staircase" behavior is projected onto two singular vectors exhibits spherical micro-clusters (referred to as a "pearl" pattern) in its spectral-subspace plot. These patterns are utilized for discovering other bot-followers using "lockstep" propagation, which has linear complexity.

### Algorithm

**Input:** Who-knows-who adjacency matrix $A$
**Output:** Set of "lockstep" followers
**Feature utilized:** Follower/followee relationships

1. Identify "seed" followers
   (a) Create a spectral subspace plot of the top $k$ left-singular vectors (left-singular vectors referring to the columns of the $U$ matrix of SVD decomposition).
   (b) Transform the points to polar coordinates ($r$ and $\theta$) using the Hough transformation.
   (c) Divide the $r$ and $\theta$ axes into bins, count the frequency, and create a histogram.
   (d) Categorize the subplot as non-lockstep, non-overlap lockstep, or lockstep. If the $\theta$ bin plots show two apparent spikes at 0 and 90 degrees, then there is a non-overlap lockstep behavior (aka "ray" pattern). If the $r$-bin plot shows a single spike apart from $r = 0$, then there is an overlap lockstep behavior present (aka "pearls" pattern). Spikes can be detected using median filtering and identifying the related nodes as "seeds".
2. Lockstep propagate
   (a) For each "seed" followee identified, count the number of followers in the seed set. If there are more than a certain threshold, then the group is considered "lockstep"

followees.

(b) For each follower, count the number of its followees that are considered "lockstep". If above a certain threshold, then determine the follower as part of the "lockstep" behavior. From this step, new "lockstep" followers are determined while other followers are considered benign because they only have zero or one "lockstep" followee.

(c) Repeat (a) and (b) until convergence.

#### Results

While the lockstep method was able to detect bots that try to camouflage themselves by following legitimate users different from its bot group, it relies on the assumption that bots use most of their activity to follow a followee as a mass.

### Modified COMPA – 2015

#### Approach

The authors suggest a modified version of the COMPA algorithm in order to detect compromised social media accounts. Often, online social networks contain automated bot accounts for malicious reasons. Currently, there are many algorithms for detecting anomalies but not many for detecting hijacked accounts. Hijacking accounts is becoming more popular for malicious users because a new bot account does not have to be created. Instead, the spammer will take control of a pre-existing account with a good reputation. The original COMPA system detects these types of accounts by clustering similar messages together. Although the original COMPA system can accurately detect large-scale spam campaigns, it is hard for it to detect individual hijacked accounts. The authors suggest that they can modify COMPA to detect individual hijacked accounts by removing the clustering part of the algorithm.

#### Model

Previous models that were used before COMPA included the semi-supervised learning framework SybilBelief, the unsupervised approach of principal component analysis, and the support vector machine. However, these algorithms cannot pinpoint exactly when a particular account was hijacked. COMPA is a supervised learning classifier that classified based on each user account. If a new tweet is anomalous, then the account is considered to be hijacked.

For this modified version of COMPA, the features are a list of tuples containing the attribute value and the number of times the attribute occurs. To smooth out these features, the algorithm looks at the previous and next hour as well. From this information, the model is trained and anomaly scores are calculated from the new messages. The overall anomaly score is calculated from the weighted sum of the individual anomaly scores. To evaluate this model, the data is collected from a number of social media accounts where a subset of the messages are swapped with anomalous messages. This way, the authors have full control over when the hijackings of the account took place.

#### Results

While the modified COMPA model is able to detect individual hijacked accounts, it has some limitations. There is a tradeoff between precision and recall. If the precision is too low, it cannot be used in real world situations. However, if the anomaly threshold is not set to a high value, there could be too many false positives. The model can also be unstable in that the detection behavior is dependent on the threshold. In the future, the authors suggest that the model could be improved by adding more features to the training data.

## EdgeCentric: Anomaly Detection in Edge-Attributed Networks – 2015

### Approach

Many real-world online interactions can be represented using graphs with attributed-edges. The attributed edges in graphs capture information about how the adjacent nodes interact with other entities in the graph. For example, edges in e-commerce networks indicate how and when users rated products; edges in unipartite social networks contain information about when friendships were formed and when users communicated with each other.

EdgeCentric is an information theoretic approach for node anomaly detection in edge-attributed graphs because it uses the information on edges to identify suspicious nodes. It leverages minimum description length (MDL) to identify nodes with atypical behavior on adjacent edges in an unsupervised fashion.

### Model

There are multiple types of nodes, relations, and attributes. For example, in the e-commerce networks: nodes can be of type products, buyers or sellers; relations can be of type "users rated products" or "users rated sellers"; attributes can be the time-stamps or the actual value of ratings. Also, there can be multiple model distributions, forming clusters of typical node behavior.

The main idea of the paper is to devise an abnormality function, $\delta(v)$, corresponding to each node $v$ in the edge-attrbuted multi-graph. The higher the value of $\delta(v)$, the more likely the node is suspicious. $\delta(v)$ is formulated as:

$$\delta(v) = \sum_{r \in R} \left( \sum_{w \in \Omega(r)} \left( |f_{v,r}| \sum_{g=1}^{h_w} (\rho_{b,r,w,g}.KL(\hat{v}_w || C_{b,r,w,g})) \right) \right)$$

where $|f_{v,r}|$ is the cardinality of the edge-attribute value vector $f_{v,r}$ generated from $v$'s neighboring edges of type $r$, $\hat{v}_w$ is the discrete probability distribution corresponding to $v$ over attribute $w$, $C_{b,r,w,g}$ and $\rho_{b,r,w,g}$ give the $g^{th}$ model distribution and proportion of the $g^{th}$ cluster on the $r^{th}$ relation type, respectively. $KL(\dot{||}\dot{)}$ is the KL divergence in bits defined on two data models. Intuitively, $\delta(v)$ gives the expected number of extra bits required to encode the node $v$s edge-attribute vectors with respect to a joint model over multiple relations, attributes and clusters.

### Algorithm

The algorithm is split into five steps:

1. Aggregation: For each node-type, aggregate attribute values over the outgoing edges from each node for each associated relation-type. For each node, there will be a vector of attribute values for each relation-type.
2. Discretization: Based on attribute type and range of values, discretize the space linearly or logarithmically for numerical attributes. Categorical attributes are already discretized. Normalize to construct probability mass functions.
3. Clustering: For each node-type and attribute, cluster the vectors describing the per-attribute probability mass functions associated with each relation.
4. Scoring: For each node-type, compute the abnormality function $\delta$ for all nodes over associated relations and attribute clusters.

5. Ranking: For each node-type, sort in descending order the resulting abnormality scores and return with node with scores.

**Results**

EdgeCentric is able to account for multiple relationships between entities, has practical scalability, and has numerous real-world applications because it is a graph based approach. However, it makes a lot of independency assumptions. For example, multiple attributes are assumed to be independent while formulating the closed form expression of $\delta$.

## FocusCo: Focused Clustering and Outlier Detection in Large Attributed Graphs – 2014

**Approach**

Unlike EdgeCentric, FocusCo is based on node-attributed graphs, rather than edge-attributed graphs. The existing techniques of clustering and outlier detection in node-attributed graphs do not incorporate user preference. They either use all attributes or perform unsupervised feature selection. Users cannot control the relevance of the attributes and therefore, have no control on the resultant clusters. In attributed graphs, focused clustering is often of particular interest, where users might be concerned with a few attributes. As different subsets of attributes yield different clusters, the users should be able to steer the clusters accordingly.

**Model**

Given a node-attributed graph $G(V, E, F)$ with $|V| = n$ nodes and $|E| = m$ edges, where each node has $|F| = d$ attributes, the user $u$ wants to extract only the clusters of interest. To do so, a small set $C_{ex}$ of exemplar nodes that is considered to be similar to the type of nodes the clusters of interest should contain is provided. The main idea of the paper is to infer the implicit weights $\beta_u$ (i.e., relevance) of attributes that define the nodes in $C_{ex}$. Thus, $\beta_u$ tends to be a sparse vector with large weights for only a few attributes, which are called the "focus attributes".

Once the $\beta_u$ is inferred, the focused clusters $C$ are extracted. The clusters are (1) dense in graph structure and separated from rest of the graph (2) coherent on focused attributes with large weights. The focused clusters can be overlapping, sharing several of their nodes. The other objective is to detect outliers $O$, which are defined to be the nodes that structurally belong to a focused cluster, but deviate from its members in some focus attributes.

**Algorithm**

The three main components of FocusCo are:

1. *Inferring attribute weights*: The user initiates the focused clustering by providing $C_{ex}$. The goal is to identify the relevance weights that make nodes in $C_{ex}$ similar to each other (i.e. they have small Mahalanobis distance to each other). Mahanalobis distance between two feature vectors $f_i$ and $f_j$ is defined as $(f_i - f_j)^T \beta_u (f_i - f_j)$. For this, the following optimization problem is solved:

$$\min_{A} \sum_{(i,j)\in P_S} (f_i - f_j)^T \beta_u (f_i - f_j) - \gamma \left( \sum_{(i,j)\in P_D} \sqrt{((f_i - f_j)^T \beta_u (f_i - f_j))} \right)$$

2. *Focused Cluster Extraction*: The primary idea to create focused clusters is to first find good candidate nodes and then expand around those nodes to find the clusters. First,

the edges $E$ are re-weighted by weighted similarity of their end nodes, and the graph $G$ is induced on the edges having large weights. The nodes in the resulting connected components are considered as the candidate nodes. Such components are called core sets. Next, new nodes that increase the quality of clusters are chosen to be included in the clusters until there exist no such nodes. The quality of the a cluster $C \subset V$ is measured as its weighted conductance defined as:

$$\phi_{C,G}^{(w)} = \frac{W_{cut}(C)}{W_{Vol}(C)} = \frac{\sum_{(i,j)\in E, i\in C, j\in V\setminus C} w(i,j)}{\sum_{i\in C}\sum_{j:(i,j)\in E} w(i,j)}$$

The lower the conductance of a cluster, the better its quality.

3. *Focused Outlier Detection*: A focused cluster outlier is a node that has many edges to the members of the cluster, but deviates from the members significantly in some focus attributes. The idea is to identify the structural nodes that are best in terms of unweighted conductance during the expansion of clusters. The best nodes which were not included in the resulting focused cluster are then taken as the outliers.

**Results**

FocusCo is a scalable algorithm which runs in time linear with the size of input graph. With appropriate initializations, it can very well run in sub-linear time in practice. This helps the users to obtain clusters and identify outliers according to the attributes of choice.

## Discovering Opinion Spammer Groups by Network Footprints – 2015

**Approach**

This paper proposes an unsupervised and scalable approach for detecting opinion spammer groups based solely on network footprints. The idea is to quantify the statistical distortions caused by spamming activities in well-understood network characteristics. This graph-based measure is known as Network Footprint Score (NFS). The group spammers are identified by using a hierarchal clustering on the induced subnetwork of highly suspicious products and reviews, which outputs a set of spammers with their nested hierarchy.

**Model**

To design the NFS measure, the method uses two observations associated with real-world networks:

1. *Neighbor diversity*: The neighbors of a node should consist of nodes with diverse behavior or importance. The importance of a node is expressed in terms of its degree centrality and page-rank centrality. Both centrality measures are binned logarithmically into buckets, and their discrete probability distributions and Shannon entropies are found. The lower the entropies, the more likely the product is a target of a spam campaign.

2. *Self-similarity*: This implies that the centrality distribution of reviewers for a particular product should be similar to the overall centrality distribution of the reviewers. For each centrality measure, the self-similarity is measured in terms of the KL divergence between distributions of the reviewers of the product with the global distributions. The larger these scores, the more likely the product is the target of a spam campaign.

From the above two observations, four suspiciousness scores are obtained. The NFS score unifies all these to produce a single measure of suspiciousness. Products with high NFS values are more likely to be targets of spam campaigns.

Once the NFS values are computed for the products, the next goal is to detect the spammer groups. First, a subnetwork is constructed. It consists of products with high NFS values, the reviewers of these products, and all the products that these reviewers have reviewed. This subnetwork is represented as an adjacency matrix where each column represents a reviewer. The objective is to chip off groups where columns in a group have strong similarity.

An agglomerative clustering approach is adopted where columns are iteratively merged to form larger groups. To make the clustering more efficient, the authors use Locality-Sensitive Hashing (LSH) for finding similar sets of clusters. LSH ensures that similar points are highly likely to get hashed to the same bucket and dissimilar points are rarely hashed to the same bucket, which speeds up the similarity search.

### Results

The paper reveals that language and behavior patterns are easier to be mimicked by the spammers compared to the network-wide patterns Thus, the method utilizes these patterns for detecting spam groups. Since the proposed algorithm extensively uses various probability distributions, it is not very effective when there are only a few reviewers because a small sample of data points cannot constitute a reliable distribution.

Table 1: Summary of anomaly detection methods

| | | Input features | | | | | | Anomalies detected | | | | Others | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Year | CON | CAT | SE | SP | GR | IID | PA | CXA | COA | GA | SU | BC | SC |
| BIRDNEST | 2015 | ✓ | - | - | ✓ | - | ✓ | - | ✓ | ✓ | - | - | - | ✓ |
| Lockstep | 2014 | - | ✓ | - | - | ✓ | ✓ | - | - | ✓ | ✓ | - | ✓ | ✓ |
| PCA | 2014 | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | - | - | - | - | ✓ | ✓ |
| SFP | 2013 | ✓ | - | ✓ | - | - | - | ✓ | - | - | - | - | ✓ | ✓ |

**Input features:** CON, continuous; CAT, categorical; SE, optimal for sequential features; SP, optimal for spatial features; GR, optimal for graph features; IID, assumes i.i.d. variables

**Anomalies detected:** PA, point anomalies; CXA, context anomalies; COA, collective anomalies; GA, graph anomalies

**Others:** SU, supervised (else assume unsupervised); BC, binary classification (else assume scores); SC, scalable

## Conclusion

aaa

### Temporal Approach

aaa

### Bayesian Approach

aaa

### Group Approach

aaa