Bad Actors in Social Media

Francesca Spezzano

Boise State University

francescaspezzano@boisestate.edu

CyberSafety 2016

The First ACM International Workshop on Computational Methods for CyberSafety

Indianapolis, Oct 28, 2016

Keynote Outline

- Introduction
- Graph-based Techniques
- Behavior-based Techniques
- Hybrid Techniques

Slides available at http://bit.ly/keynote-cybersafety2016

IDENTIFYING MALICIOUS ACTORS ON SOCIAL MEDIA. Tutorial@ASONAM 2016

Srijan Kumar, Francesca Spezzano, V.S. Subrahmanian

Slides, datasets, and code: http://bit.ly/badactorstutorial

Challenges

- Little known information about bad actors/acts
- Only a small fraction of actors/acts are malicious
- Algorithm should have low false positive and false negative rates
    - Should not identify good as bad, and vice-versa
- Deal with dynamic evolving behaviors

Its like finding a needle in a haystack!

Keynote Outline

- Introduction
- Graph-based Techniques
- Behavior-based Techniques
- Hybrid Techniques

Graph-based Techniques

- Identifying bad actors by mining users' social network
  - Rank users according to centrality measures (define how important is a user within a network)
    - Degree centrality
    - Eigenvector centrality
    - Pagerank
    - HITS (Hub and Authority)

Bias and Deserve

A. Mishra et al., WWW 2011

- A vertex u's bias (BIAS) reflects the truthfulness of a node.
- Deserve (DES) reflects the expected weight of an incoming edge from an unbiased vertex.

Similarly to HITS, BIAS and DES are iteratively computed as:

$$\begin{cases} DES^{t+1}(u) = \frac{1}{|in(u)|} \sum_{v \in in(u)} [W(v,u)(1 - X^t(v,u))] \\ BIAS^{t+1}(u) = \frac{1}{2|out(u)|} \sum_{v \in out(u)} [W(u,v) - DES^t(v)] \end{cases}$$

where $X^t(v,u) = \max(0, BIAS^t(v) \times W(v,u))$.

CollusionRank

Saptarshi Ghosh et al., WWW 2012

- CollusionRank identifies link farming on Twitter
- Link farming is used by both benign and malicious users to gain influence
- CollusionRank is a pagerank-like algorithm that penalizes users who follow spammers
  - Scores range in [-1,0]

Algorithm 1 Collusionrank

Input: network, G; set of known spammers, S; decay factor for biased Pagerank, α
Output: Collusionrank scores, c
initialize score vector d for all nodes n in G

$$d(n) \leftarrow \begin{cases} \frac{-1}{|S|} & \text{if } n \in S \\ 0 & \text{otherwise} \end{cases}$$

/* compute Collusionrank scores */
$c \leftarrow d$
while c not converged **do**
  for all nodes n in G **do**

$$tmp \leftarrow \sum_{nbr \in followings(n)} \frac{c(nbr)}{|followers(nbr)|}$$

$$c(n) \leftarrow \alpha \times tmp + (1-\alpha) \times d(n)$$

  end for
end while
return c

Reduces score of known spammers

Score based on followings (and not on follower)

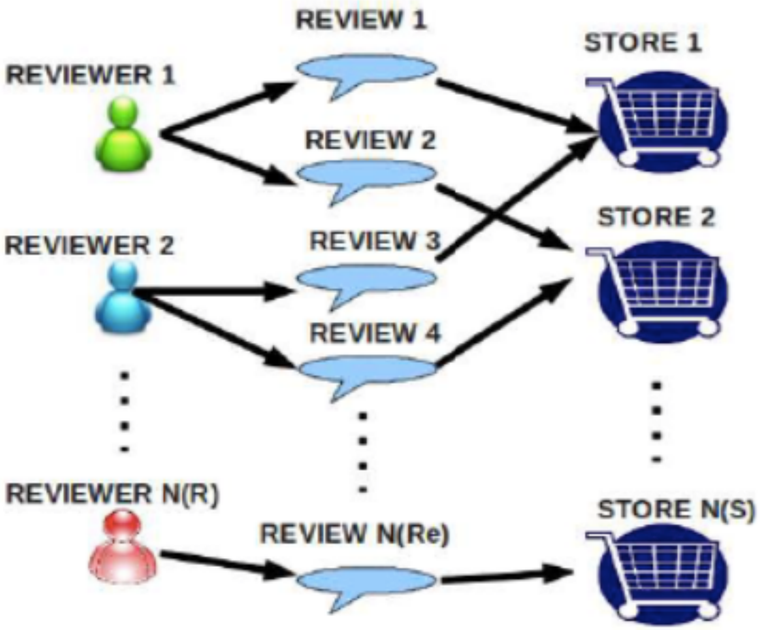
- Users with low CollusionRank score are users who are colluding with spammers
- Use CollusionRank as a filter, e.g. score users by using CollusionRank + PageRank

Store Review Spammer Detection

G. Wang et al., ICDM 2011

HITS-like algorithm to compute 3 inter-dependent measures:

- Trustworthiness of reviewer which depends (non-linearly) on its reviews' honesty scores;
- Reliability of store depending on the trustworthiness of the reviewers writing reviews for it and the score;
- Honesty of review which is a function of reliability of the store and trustworthiness of store reviewers.



CatchSync

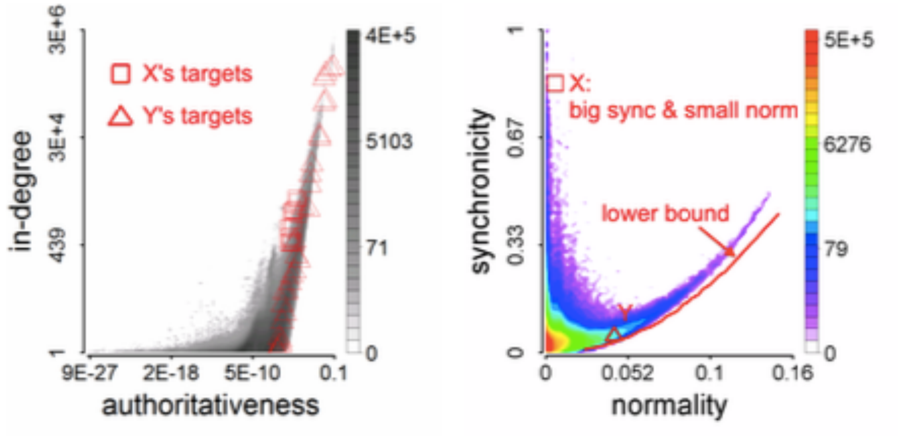
M. Jiang et al., KDD 2014

Suspicious nodes are:

- Synchronized: they connect to the very same set of nodes
- Abnormal: they behave differently from majority of the nodes
  - Node u's targets have two features: in-degree and authoritativeness

$$sync(u) = \frac{\sum_{(v,v')\in \mathcal{O}(u)\times\mathcal{O}(u)} c(v,v')}{d_o(u)\times d_o(u)}$$

$$norm(u) = \frac{\sum_{(v,v')\in \mathcal{O}(u)\times\mathcal{U}} c(v,v')}{d_o(u)\times N}$$



Suspicious nodes are the outlier in the normality-synchronicity plot

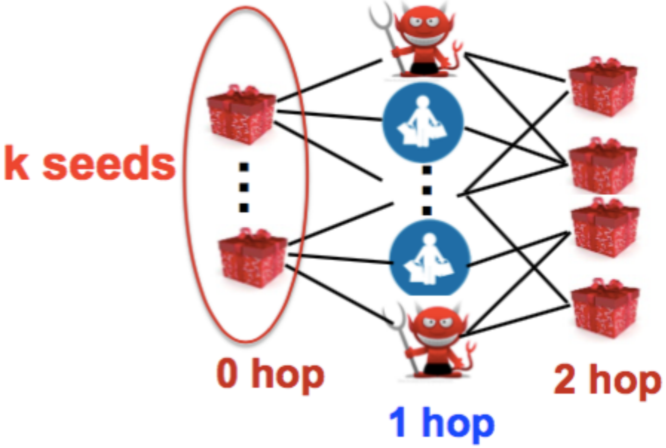
Discovering Opinion Spammers

Junting Ye et al., ECML-PKDD 2015

- Discovering spammer groups and their targeted products.
- Uses the product-review bipartite graph.

Framework consists of two components:

- Network Footprint Score (NFS): graph-based measure to quantify spammers' diversity from normal users. NFS leverages two real-world network properties: *neighbor diversity* and *network self-similarity.*
- GroupStrainer: spammers clustering algorithm on a 2-hop subgraph induced by top NFS products



Graph-based Techniques

Case studies:

- Detecting bad actors in signed networks
- Identifying nuclear proliferators via social network analysis

CASE STUDY 1:
IDENTIFYING TROLLS ON SLASHDOT

Accurately Detecting Trolls in Slashdot Zoo via Decluttering.

Srijan Kumar, Francesca Spezzano, V.S. Subrahmanian

ASONAM 2014 (https://cs.umd.edu/~srijan/trolls/)

Application: Troll Detection

Malicious users interrupt the normal functioning of online and collaborative social networks.

- Trolls
  - Users who deliberately make offensive or provocative online postings with the aim of upsetting someone or receiving an angry response.
  - Being annoying on the web, just because you can.

Example Trolling Activity

Source: www:thisisparachute.com/2013/11/trolling/

Application: Troll Detection

- Model the social network as a signed social network
- Many real SN are signed:
  - Epinion (who trusts whom on an online product rating site)
  - Slashdot (a user *u* can mark a user *v* as friend or foe)
  - Youtube (a user *u* can mark a video posted by *v* with a thumbs up or thumbs down)
  - Stack Overflow (users can mark other users' comments as good or bad)
- Past work: Rank users according to a centrality measure C
  - Identify bottom-k users as malicious users

User Ranking: Centrality Measures in SSNs

Degree-like Centrality Measures

- Freaks Centrality

$$Freaks(u) = \sum_{v \in V | W(v,u) < 0} W(v,u)$$

- Fans Minus Freaks (FMF)

$$FMF(u) = \sum_{v \in V | W(v,u) > 0} |W(v,u)| - \sum_{v \in V | W(v,u) < 0} |W(v,u)|$$

- Prestige

$$Prestige(u) = \frac{\sum_{v \in V | W(v,u) > 0} |W(v,u)| - \sum_{v \in V | W(v,u) < 0} |W(v,u)|}{\sum_{v \in V | W(v,u) > 0} |W(v,u)| + \sum_{v \in V | W(v,u) < 0} |W(v,u)|}$$

User Ranking: Centrality Measures in SSNs

Pagerank/eigenvector-like Centrality Measures

- Pagerank $$PR(u) = \frac{1-\delta}{|V|} + \delta \sum_{v \in pred(u)} \frac{PR(v)}{|succ(v)|}$$
- Modified Pagerank: Mod-PR(u) = $PR^{+}(u) - PR^{-}(u)$
- Signed Spectral Rank (SSR): Pagerank of the signed adjacency matrix *A*
- Negative Rank (NR): NR(u)=SSR(u) – PR(u)
- Signed Eigenvector Cerntrality (SEC): is the vector *x* that satisfies the equation $Ax = \lambda x$

User Ranking: Centrality Measures in SSNs

Modified HITS

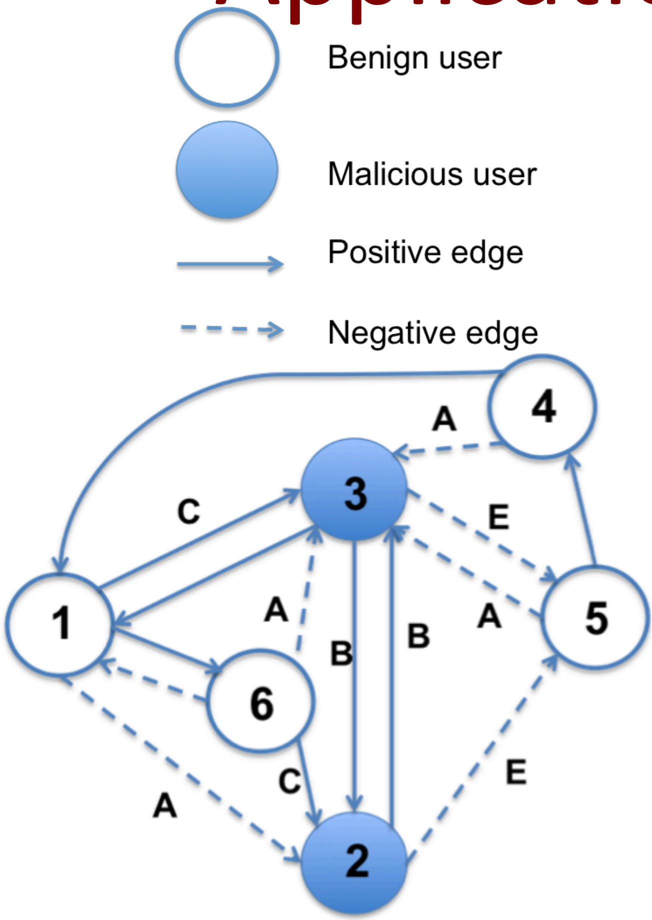
Iteratively computes the hub and authority scores separately on A^+ and A^-, using the equations:

$$\begin{cases} h^+(u) = \sum_{v \in out^+(u)} a^+(v); \; a^+(u) = \sum_{v \in in^+(u)} h^+(v) \\ h^-(u) = \sum_{v \in out^-(u)} a^-(v); \; a^-(u) = \sum_{v \in in^-(u)} h^-(v) \end{cases}$$

Then assign $h(u) = h^+(u) - h^-(u)$

and $a(u) = a^+(u) - a^-(u)$

Application: Troll Detection



Measure	Lowest					Highest
Freaks	**3**	5	1,2		4,6	
FMF	5	**3**	1,2,4,6			
Prestige	5	**3**	1,2		4,6	
M-PR	5	**3**	4	6	1	2
SSR	5	4	6	1	2	3
NR	5	4	1	6	2	3
SEC	5	4	6	1	2	3
M-HITS	**3**	5	4	6	1	2
BAD	5	**3**	2	1	4,6	

TIA: Troll Identification Algorithm

IDEA

– Remove the “hay” from the “haystack”, i.e. remove irrelevant edges from the network, to bring out interactions involving at least one malicious user.

– Then find the “needle” in the reduced “haystack”.

Kumar S, Spezzano F, Subrahmanian VS. *Accurately detecting trolls in slashdot zoo via decluttering*. In IEEE/ACM ASONAM, 2014

TIA: Troll Identification Algorithm

INPUT: A SSN G, a centrality measure $\mathcal{C}$, a threshold τ, and a set S of decluttering operations
OUTPUT: A score for the nodes

Three steps in the algorithm:

1. Use $\mathcal{C}$ (and τ) to tentatively mark users as benign or malicious.
2. Declutter the graph by removing interactions among the found benign users.
3. Iterate 1-2 till no more edges can be removed.

Decluttering Operations

Given a centrality measure *C*, we mark as **benign**, users with centrality score greater than or equal to a threshold **τ**. The remaining users are marked **malicious**.

Definition (Decluttering Operation)

A *decluttering operation* is an associative function $\rho : \mathcal{G} \to \mathcal{G}$ that transforms graphs into graphs such that for all $G = (V, E, W)$, if $\rho(G) = G' = (V', E', W')$, then $V = V'$, $E' \subseteq E$, and for all $e' \in E'$, $W'(e') = W(e')$.

Between benign nodes:

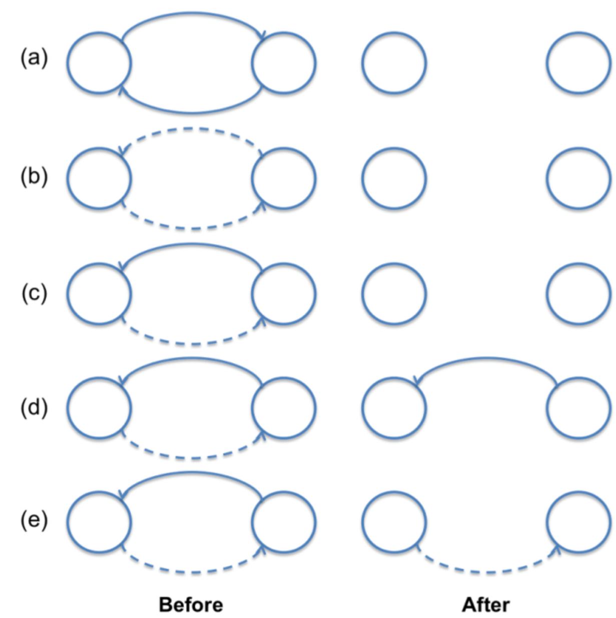
(a) Remove positive edge pairs

(b) Remove negative edge pairs

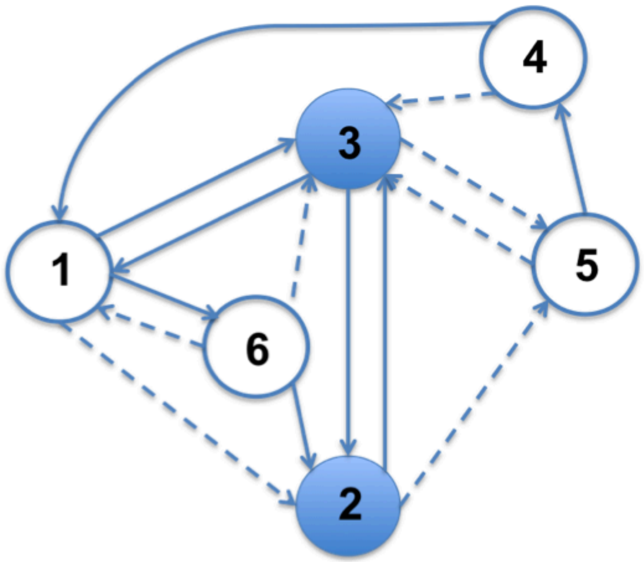
(c) Remove positive-negative edge pairs

(d) Remove negative edge in positive-negative edge pairs

(e) Remove positive edge in positive-negative edge pairs



TIA Example

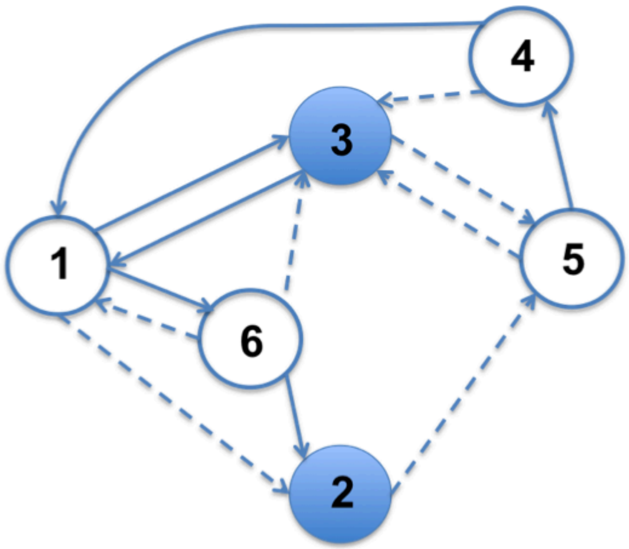


Decluttering Operations:
(a) Remove positive edge pairs
(b) Remove negative edge pairs
(d) Remove negative edge in positive-negative edge pairs
Threshold τ=0

Nodes	1	2	3	4	5	6
NR	-0.06	0.13	0.45	-0.40	-0.68	-0.01

Figure : TIA algorithm iteration 1 by using Negative Rank and DOP = {a,b,d}

TIA Example

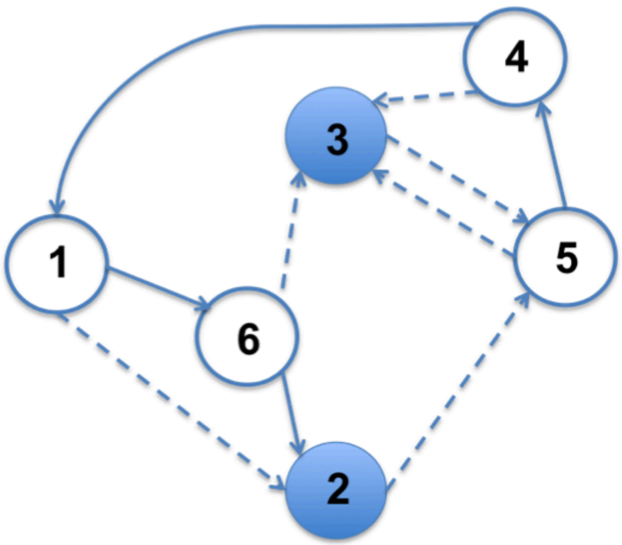


Decluttering Operations:
(a) Remove positive edge pairs
(b) Remove negative edge pairs
(d) Remove negative edge in positive-negative edge pairs
Threshold τ=0

Nodes	1	2	3	4	5	6
NR	0.02	-0.15	0.48	-0.49	-0.72	0.04

Figure : TIA algorithm iteration 2 by using Negative Rank and DOP = {a,b,d}.

TIA Example



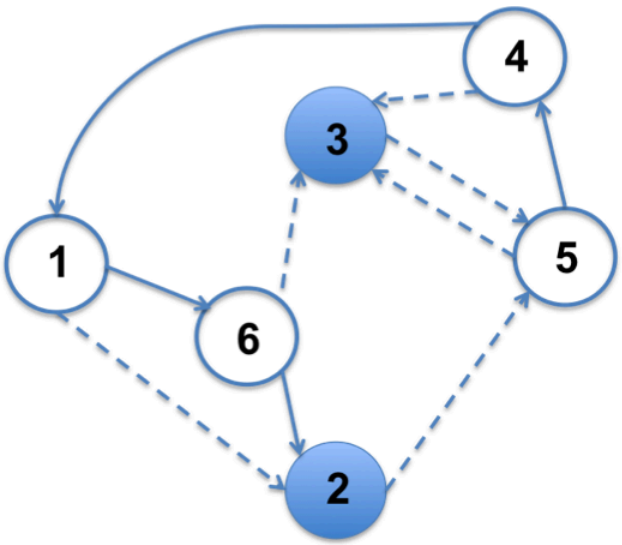
Decluttering Operations:
(a) Remove positive edge pairs
(b) Remove negative edge pairs
(d) Remove negative edge in positive-negative edge pairs
Threshold τ=0

Nodes	1	2	3	4	5	6
NR	0.13	-0.11	-0.85	0.21	0.34	0.07

Figure : TIA algorithm iteration 3 by using Negative Rank and DOP = {a,b,d}.

No more decluttering operations are possible

TIA Example



Decluttering Operations:
(a) Remove positive edge pairs
(b) Remove negative edge pairs
(d) Remove negative edge in positive-negative edge pairs
Threshold τ=0

Nodes	1	2	3	4	5	6
NR	0.13	-0.11	-0.85	0.21	0.34	0.07

Figure : TIA algorithm iteration 3 by using Negative Rank and DOP = {a,b,d}.

Result: 1,4,5 and 6 are benign, 2 and 3 are malicious

Experiments

- **Dataset**: we tested our TIA algorithm on Slashdot
  - Technology-related news website.
  - Contains threaded discussions among users.
  - Comments labeled by administrators
    - **+1** if they are normal, interesting, etc. or
    - **-1** if they are unhelpful/uninteresting.
  - There are 71.5K nodes and 490K edges (24% negative).
  - Ground truth available (96 users marked as trolls by Admin account).

Experiments

Best Settings

Centrality	*None*	*a,c*	*a,e*
Freaks	15.07	14.77	15.22
FMF	3.13	4.35	4.64
Prestige	0.18	0.2	0.2
M-PR	1.25	0.94	1.12
SSR	10.27	-	-
NR	13.9	-	-
SEC	3.42	50.96	51.04
M-HITS	13.38	15.79	15.88
BAD	0.18	0.19	0.19

Table comparing Average Precision (in %) using TIA algorithm on Slashdot network

(Original + Best 2 columns only)

Average Precision is the area under the Precision-Recall curve

Centrality	*None*	*a,c*	*a,e*
Freaks	17	16	17
FMF	10	10	10
Prestige	0	0	0
M-PR	6	6	7
SSR	19	-	-
NR	24	-	-
SEC	7	51	51
M-HITS	0	0	0
BAD	0	0	0

Number of Trolls (out of 96)

Average Precision of random ranking is 0.001%

We retrieved more than twice as many trolls as NR

Experiments

Measure	95%		90%		85%		80%		75%	
	MAP	Runtime	MAP	Runtime	MAP	Runtime	MAP	Runtime	MAP	Runtime
Freaks	15.35%	0.17	15.22%	0.17	15.12%	0.16	15.46%	0.15	15.62%	0.14
FMF	3.23%	0.24	3.16%	0.26	3.2%	0.23	3.52%	0.21	3.42%	0.19
Prestige	0.18%	0.34	0.18%	0.36	0.18%	0.31	0.19%	0.29	0.19%	0.26
M-PR	1.31%	12.6k	1.3%	10.9k	1.43%	8.9k	1.67%	8.3k	1.6%	7.6
SSR	10.34%	1.7k	10.27%	1.6k	10.21%	1.4k	9.95%	1.2k	10.05%	1.1k
NR	13.66%	2.2k	13.45%	1.9k	13.38%	1.7k	13.08%	1.6k	13.21%	1.4k
SEC	3.27%	5.21	3.3%	4.75	3.27%	4.29	3.56%	3.97	3.27%	3.6
M-HITS	13.65%	27.96	13.17%	25.84	13.29%	24.37	13.73%	23.71	14.66%	22.09
BAD	0.18%	32.55	0.18%	29.97	0.19%	27.11	0.19%	24.15	0.2%	21.9
SEC + *a*,*c*	51.14%	47.75	51.33%	43.79	51.02%	43.53	52.14%	35.33	51.14%	39.64
SEC + *a*,*e*	51.24%	46.87	51.4%	42.9	51.12%	42.8	52.22%	33.12	51.24%	37.68

Table showing running times (in sec.) and Average Precision averaged over 50 different versions for 95%, 90%, 85%, 80% and 75% randomly selected nodes from the Slashdot network.

We are 3 times better than Freaks in MAP
The running time is less than 1 min.

CASE STUDY 2: IDENTIFYING NUCLEAR PROLIFERATORS VIA SOCIAL NETWORK ANALYSIS

SPINN: Suspicion Prediction in Nuclear Networks

Ian Andrews, Srijan Kumar, Francesca Spezzano, V.S. Subrahmanian

IEEE Intelligence and Security Informatics (ISI), 2015

SPINN: Suspicion Prediction in Nuclear Networks

- Given a network with some nodes marked as “good” and some as “bad,” predict which nodes in a Nuclear Proliferation Network (NPN) are suspicious.

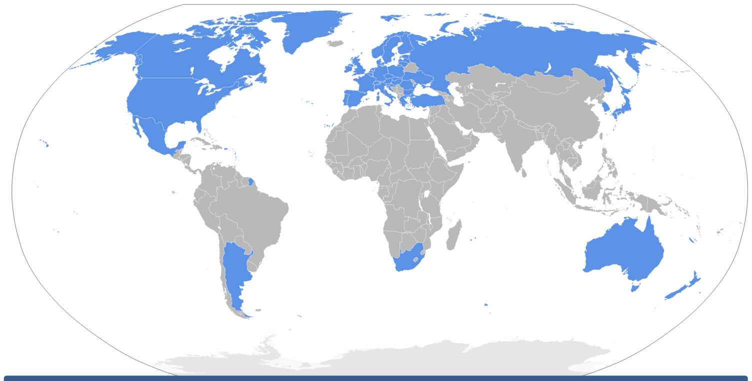
- We developed the largest (to the best of our knowledge) network related to nuclear non-proliferation.

The SPINN Dataset

- Overall dataset consisted of 74,060 entities (companies, agencies, and people) and 1,091,005 edges, or relationships between entities
- Weighted network consisting of three components:
  - Blacklist *(Known proliferators)*: entities mainly gathered manually from data in the US Department of Treasury list of Specially Designated Nationals (SDN)
  - Whitelist
  - Unknown

The SPINN Dataset

OFAC



Wassenaar Agreement

LinkedIn

Bloomberg

Suspicious Node Prediction: Features

- Variables needed to help determine which "unknown" nodes were more likely to be suspicious
- Node properties important, but not sufficient
- Characteristics of the relationships between nodes must be exploited

Node properties

- ***Country suspicion score***
  - 1-10 score calculated using Corruption Perception Index rank, sanctions status, and NPT treaty and Waasenaar Arrangement status
- ***Name suspicion***
  - Drawn from keywords matched to name of entity
  - A company with the words “mining” or “nickel” more likely to be nuclear-relevant than a clothing retailer
- ***Specialty suspicion***
  - A set of suspicious specialties is maintained, and compared with the specialty of the entity in question
  - For example, a nuclear scientist is more likely to earn a high suspicion score based on this metric than a surgeon.

Network properties

- Several network properties were defined and implemented in Java using the SPINN dataset:
  - Number of nearby suspicious neighbors
  - Number of nearby non-suspicious neighbors
  - Distance to closest suspicious node
  - Distance to closest non-suspicious node
  - Number of neighbors with suspicious specialties
  - Number of suspicious specialties among neighbors

Defining Suspiciousness Rank

- Suspiciousness Rank SR(u) is a comprehensive rank based on the Pagerank algorithm
  - SR builds on PageRank by considering blacklisted and whitelisted nodes
  - Suspiciousness rank of a node will increase with that of its neighbors
- Implemented in two variations: with and without bias

Defining Suspiciousness Rank (cont'd)

$$SR(u) = (1-d)\sum_{w \in V_P \cup V_O} SR(w)I(w) + d\left[\sum_{(v,u,ep)\in E} \frac{SR(v)\omega(v,u,ep)}{\sum_{(v,u',ep')\in E}\omega(v,u',ep')}\right] \quad (1)$$

- *I(w)* can be used to adjust the level of bias introduced by a node's suspicion value
- *d* is a damping factor set to 0.85 (as in Pagerank)

Suspiciousness rank with bias

- In our dataset, there are fewer suspicious than non-suspicious nodes, so the bias for suspicious nodes is higher than unknown
- *I(w)* is defined as follows:

$$I(w) = \begin{cases} 0, & \text{if } w \text{ is NonSuspicious.} \\ 1/(2 \times \#SuspiciousNodes), & \text{if } w \text{ is Suspicious.} \\ 1/(2 \times \#UnknownNodes), & \text{if } w \text{ is Unknown.} \end{cases}$$

Implementation

- Each of these features computed in a 10-fold cross-validation experiment
  - 90% of the whitelist and blacklist used as training data; balance used to test classifier accuracy
- Matthews Correlation Coefficient (MCC) chosen due to robustness and applicability when class sizes are disparate

Results

Classifier	Mean MCC	MCC Std. Dev.
Random Forest (10 trees)	0.867	0.11
Gaussian Naive Bayes	0.846	0.018
SVM(rbf, C=0.01)	0.854	0.025
SVM(linear, C=0.01)	0.956	0.015

- SVM with linear kernel had the highest mean MCC value and a low standard deviation
- SVM is able to distinguish suspicious nodes with high consistency

SPINN: real-world applications

- Has been used to identify previously unknown suspicious entities
- Example: A Malaysian electronics fabricator
  - 20th most suspicious country out of 177
  - Applications include metal processing, plastics, Chemical engineering
  - Substantial distribution network that spans several other suspicious countries (incl. Iran, Pakistan, Syria)
  - reprimanded for violating market listing requirements
  - Shares at least one banking connection with a company identified as part of the AQ Khan network

Effective in real world!

Keynote Outline

- Introduction
- Graph-based Techniques
- Behavior-based Techniques
- Hybrid Techniques

Behavior Models

Behavior models are aspects of users as portrayed by its interactions with other users and information, in terms of certain properties.

User to user interaction:

– Friend, Follow, Enemy

User to information interaction:

– Comment, Like, Dislike, Upvote, Downvote

Properties:

- Timestamp
- Count
- Distribution
- Importance
- Centrality
- Popularity, etc.

Behavior Models

How to model behaviors? E.g. temporal behavior with timestamps?

1. Sort timestamps in increasing order
2. Calculate difference between consecutive timestamps
3. Create N bins (linear or log-scale)
4. Calculate frequency of each bin.
5. Normalize the frequency. This is the temporal behavior

Example

TS = <100, 65,20, 135, 100, 190, 175>

Sorted_TS = <20, 65, 100, 100, 135, 175, 190>

Difference_TS = < 45, 35, 0, 35, 40, 15>

Bins = [0,9], [10,19], [20,29], [30,39], [40,49]

Frequency = < 1, 1, 0, 2, 2>

Behavior_TS =
< 1/6, 1/6, 0/6, 2/6, 2/6>

Behavior Models

Given a set of interactions, how do we create behavior models to detect malicious users?

<u>Supervised</u>

1. Create behavior models of known malicious and known non-malicious actors in the same properties.

2. Create machine learning models that distinguishes between the two.

Large scale

Requires labeled data
Feature engineering

Behavior Models

Given a set of interactions, how do we create behavior models to detect malicious users?

Unsupervised

1. Create global distribution of properties of all users
2. Find users that deviate from the global distribution
   → These are suspicious/malicious

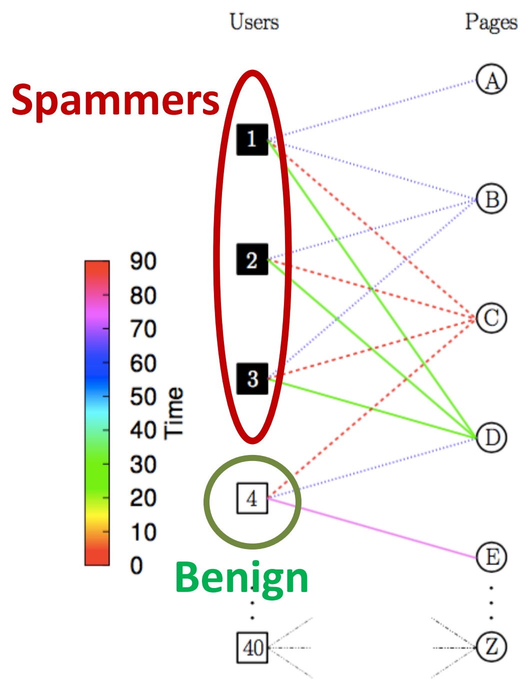
No labels required

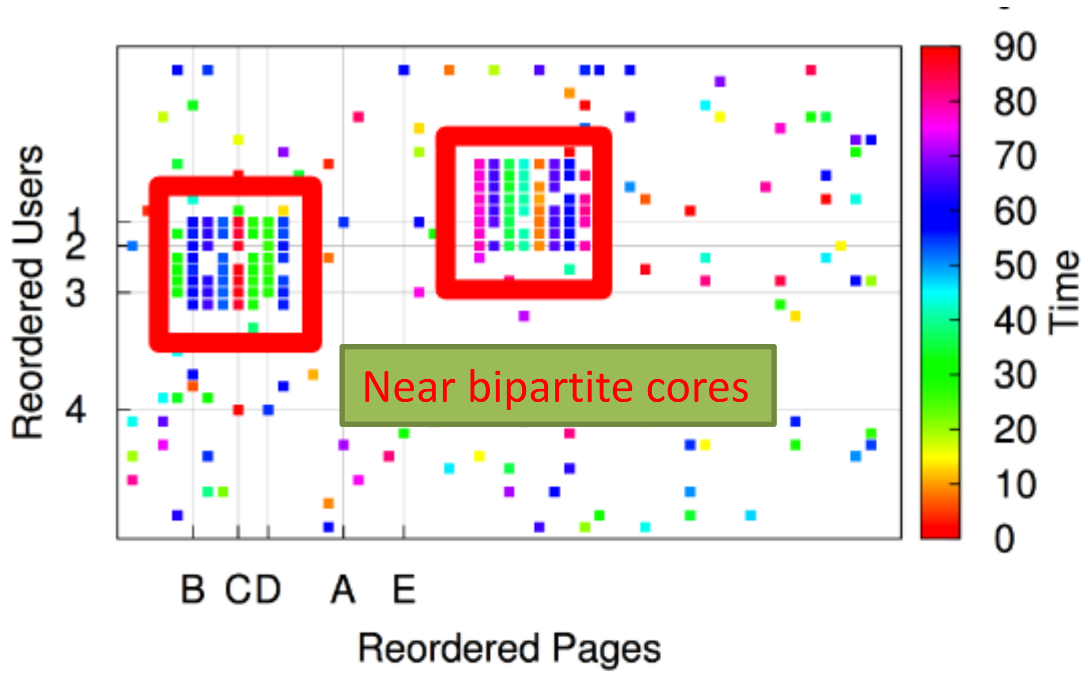
Tuning to suit needs
Computationally challenging

CopyCatch

A. Beutel et al., WWW 2013

- Identify fake likes on Facebook having lockstep pattern (liking same pages around same time)
- Unsupervised behavior model to identify dense block in a user-page-timestamp matrix

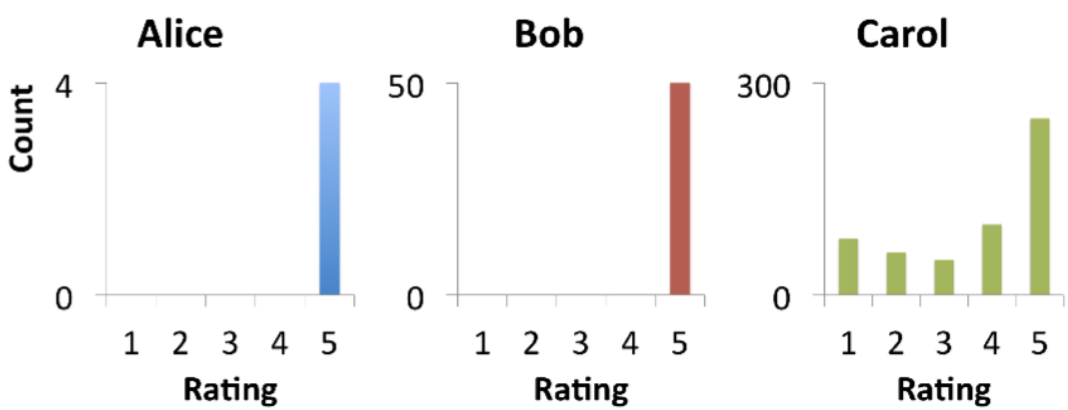


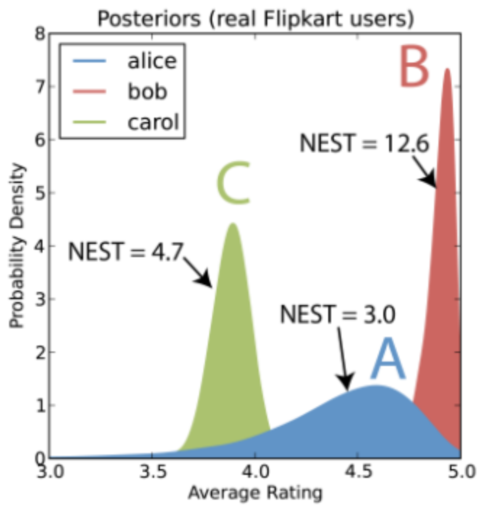


BIRDNEST

B. Hooi et al., SDM 2016

- Identify fraud in rating networks
- Fake reviews
  1. occur in short burst of time
  2. Malicious users have skewed rating distributions





- Bayesian Inference for Rating Data (BIRD) to model of user rating behavior
- Normalized Expected Surprise Total (NEST): likelihood-based suspiciousness metric (unsupervised)

Antisocial behavior

J. Cheng et al., ICWSM 2015

- Identify trolls on three comment platforms
  - *CNN.com* (general news), *Breitbart.com* (political news), and *IGN.com* (computer gaming)
- Supervised behavior model based on:
  - Post Content
  - Comment and interaction activity
  - Community feedback

Feature Set	Features
Post (20)	number of words, readability metrics (e.g., ARI), LIWC features (e.g., affective)
Activity (6)	posts per day, posts per thread, largest number of posts in one thread, fraction of posts that are replies, votes given to other users per post written, proportion of up-votes given to other users
Community (4)	votes received per post, fraction of up-votes receieved, fraction of posts reported, number of replies per post
Moderator (5)	fraction of posts deleted, slope and intercept of linear regression lines (i.e., m_1,m_2,c_1,c_2)

Behavior-based Techniques

Next invited talk

"Vandals and Hoaxes on the Web"

by Srijan Kumar

VEWS: A Wikipedia Vandal Early Warning System

Srijan Kumar, Francesca Spezzano, V.S. Subrahmanian, SIGKDD 2015

Disinformation on the Web: Impact, Characteristics, and Detection of Wikipedia Hoaxes

Srijan Kumar, Robert West, Jure Leskovec, WWW 2016

Keynote Outline

- Introduction
- Graph-based Techniques
- Behavior-based Techniques
- Hybrid Techniques

Active Methods

1. Insert a “trap” in the system to attract bad users, e.g.
   – Honeypots
   – Buying Fake Followers
2. Perform an analysis of the properties of these bad profiles for creating classifiers to actively filter out existing and new bad users.

Social Honeypots for Spam Detection

K. Lee et al. SIGIR 2010

- MySpace: 51 honeypots over 3 months
- Twitter: Unknown number of honeypots over 2 months.
- Two step process:
  - Identify accounts that friend/follow the honeypots.
  - Use an SVM classifier to distinguish between spammers and benign accounts.

K. Lee, J. Caverlee, S. Webb. Uncovering Social Spammers: Social Honeypots + Machine Learning, *Proc. SIGIR 2010*.

MySpace Spam Profiles

- Click Traps: Users clicking on objects on the profile page are redirected to another webpage.
- Infiltrators: Spams friends of those who accept a friend request.
- Pornography: "About Me" section of the profile shows porn stories and links to porn sites
- Dubious Pills: Similar to the above
- Winnies: All these profiles have the headline "Hey its winnie" even though the rest of the profile is different. Links lead to porn sites.

Understanding Facebook Like Fraud Using Honeypots

- *like farms* sell fake likers to inflate the number of Facebook page likes
- 13 Facebook *honeypot* pages were deployed to catch fake likers
- comparative analysis based
  – demographic,
  – temporal, and
  – social characteristics of the likers.

- **Findings**: likers come from specific countries, their profiles, the majority of them are male, and 2 modus operandi performed by link farms
  – Farms operated by bots
  – Farms mimicking regular users' behavior

De Cristofaro et al. Paying for Likes? Understanding Facebook Like Fraud Using Honeypots *Proc. IMC 2014*.

Uncovering Fake Likers in Online Social Networks

- Honeypot to collect *fake Likers* from *Fiverr* and *Microworkers*

- High accuracy (0.897) outperforming PCA, SynchroTrap, and CopyCatch.

Prudhvi Ratna Badri et al. Uncovering Fake Likers in Online Social Networks. *Proc. CIKM 2016*.

Content-based Features

- Analyze user posts content
  - Syntactical aspects
  - Semantics: sentiment, topics discussed
- Shared image content
  - Posted Instagram images have been used to detect cyberbullying

H. Hosseinmardi et al. Prediction of Cyberbullying Incidents in a Media-based Social Network. *Proc. ASONAM 2016*.

Social Spammer Detection with Sentiment Information (X. Hu et al. ICDM 2014)

- Used 3 datasets
  - TAMU Honeypot data 30K users (7 months) with about a 50/50 split into benign vs. spammers
  - Twitter Suspended Spammers data. ~2 mths , ~20K users with ~4K spammers
  - Stanford Twitter Sentiment. 40K tweets over 2.5 months with labeled sentiment.

X. Hu, J. Tang, H. Gao, H. Liu. Social Spammer Detection with Sentiment Information, ICDM 2014.

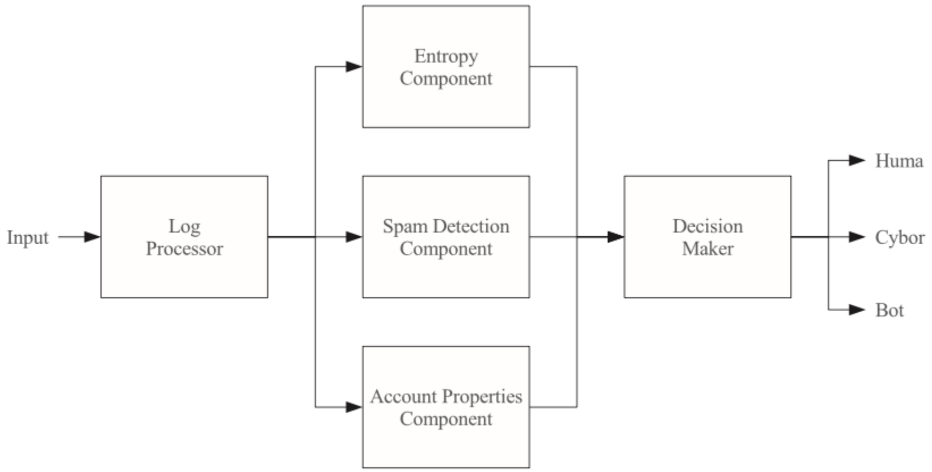
1) Associate sentiment vector *s(u)* with each user *u*. *s(u)* is the vector of sentiment for ALL tweets in the data set.
2) Defined distance between two users' sentiment vectors.
3) Shorter distance between users in same category
4) More similar sentiment vector between neighbors
5) Set up the problem of finding spammers as non-convex optimization problem
6) Develop a novel algorithm to solve this problem.

Achieve high precision and recall (over 0.9 for both) on both test datasets.

Detecting Bots/Cyborgs on Twitter

(Z. Chu et al. IEEE TDSC 2012)

- Introduces cyborgs – bot-assisted human accts or human-assisted bot accts
- Developed a training set with about 2K accounts per category (human, bot, cyborg)
- Studied the main differences between these categories.



Z. Chu, S. Gianvecchio, H. Wang and S. Jajodia. Detecting Automation of Twitter Accounts: Are you a Human, Bot, or Cyborg? IEEE Transactions on Dependable & Secure Computing, Vol 9, Nr. 6, pages 811-824, 2012

Detecting Bots/Cyborgs on Twitter

(Z. Chu et al. IEEE TDSC 2012)

	Bots	Cyborgs	Humans
Do bots have more friends than followers?	3rd	2nd	1st
Does automation generate more tweets?	3rd	1st	2nd
Does automation yield higher tweet frequency?	1st	2nd	3rd
Are bots posts more regular ?	Lowest entropy		Highest entropy
How do bots post vs. humans?	API		Twitter website
Do bots include more links in their tweets than humans?	1st	2nd	3rd

CASE STUDY 3:
IDENTIFYING BOTS ON TWITTER

Using Sentiment to Detect Bots on Twitter: Are Humans more Opinionated than Bots?
J. Dickerson, V. Kagan, and V.S. Subrahmanian.
ASONAM 2014

Dataset Creation

- 2014 Indian Election
  - Largest democratic election in history
  - Social media played huge role
- Defined set of topics of interest (TOI):
  - Political parties: Shiv Sena, BJP, …
  - Politicians: Rajnath Singh, Nitish Kumar, …
- Data from July 15 2013 to May 15 2014
- Network: Users who twitted about TOI and their 2-hops neighbors
  - 7.7M+ tweets
  - 550K+ users
  - 40M+ edges
- 897 users labeled as either bots or normal users through Mechanical Turk

Sentiment Extraction

- For each user *u*, day *d*, and topic *t*:

SS(*d*,*u*,*t*): sentiment score in [-1,+1] for topic *t* averaged across all *u*'s tweets on *t* for day *d*

- Past work did not look at *topic-specific* sentiment for detecting malicious actors
- Used SentiMetrix's commercially-available:
  - SS(*d*,*u*,*t*) = -1 → "maximally negative"
  - SS(*d*,*u*,*t*) = +1 → "maximally positive"
- Could use other methods as long as they assign a sentiment score to a topic

Features

- Tweet Syntax
  - E.g. #hashtags, #mentions, #links, etc
- Tweet Semantics
  - Lots of sentiment related features for user
- User Behavior
  - Tweet spread/frequency/repeats/geo
  - Tweet volume histograms by topic
  - Sentiment: normalized flip flops(t), variance(t), monthly variance(t)
- User Neighborhood (and behavior)
  - Multiple measures looking at agreement/disagreement between user sentiments and those of people in his neighborhood

Using Sentiment to Detect Bots on Twitter: Are Humans more Opinionated than Bots?,
J. Dickerson, V. Kagan, and V.S. Subrahmanian.
ASONAM 2014

Tweet Semantics Features

Contradiction Rank

$$CR(u,t) = x^+_t\, y^-_t + x^-_t\, y^+_t$$

- where
  - x^+_t is the fraction of *u*'s tweets with sentiment that are positive w.r.t. *t*
  - y^+_t is the fraction of all tweets [not just *u*'s] with sentiment that are positive w.r.t. *t*
  - x^-_t, y^-_t defined similarly
- High contradiction rank => most users disagree with *u* on *t*
- Low contradiction rank => most users agree with *u* on *t*

Agreement Rank:

$$AR(u,t) = x^+_t\, y^+_t + x^-_t\, y^-_t$$

Dissonance rank of user

$$\mathrm{DR}(u) = \sum_{t \in \mathrm{TOI}} \mathrm{CR}(u,t)/\mathrm{AR}(u,t)$$

Positive Sentiment Strength

- Average sentiment score (for *t*) from *u*'s tweets that are positive about *t*

+/- Sentiment Polarity Fraction

- Percentage of *u*'s tweets on *t* that are positive/negative

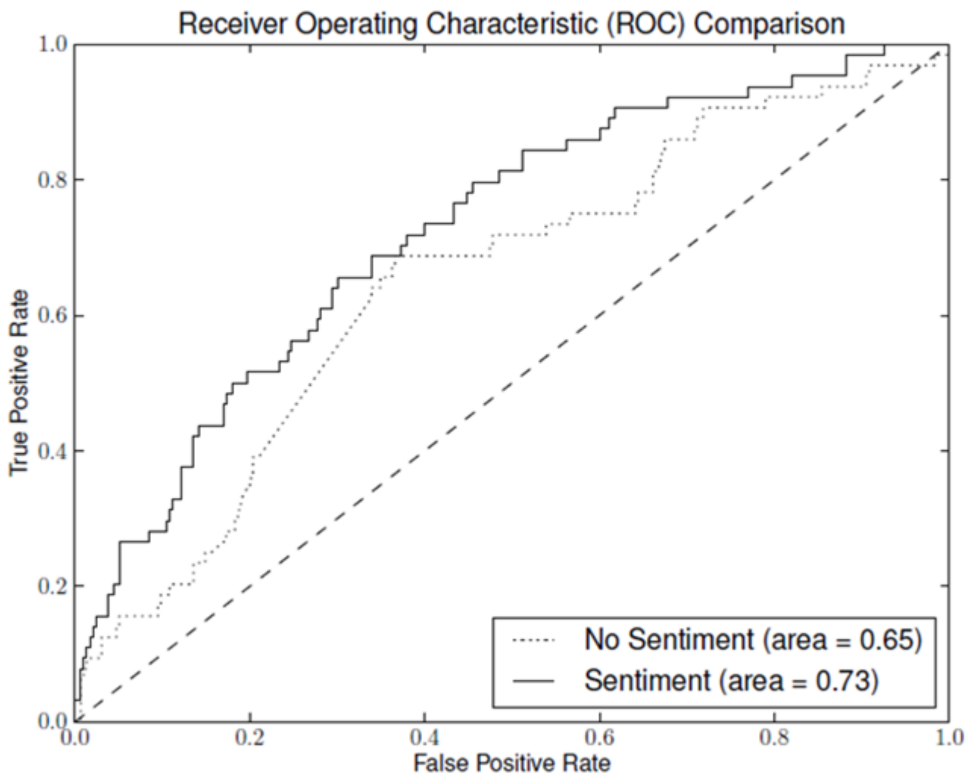
Network Features

- Neighborhood Contradiction Rank
  - Similar to contradiction rank: but y_t^+, y_t^- are computed by just considering *u*'s neighbors' tweets.

- Intuition:
  - *u*'s (global) contradiction rank could be high because *u*'s opinions on t are inconsistent with the majority view
  - But may be consistent with *u's* immediate neighborhood.

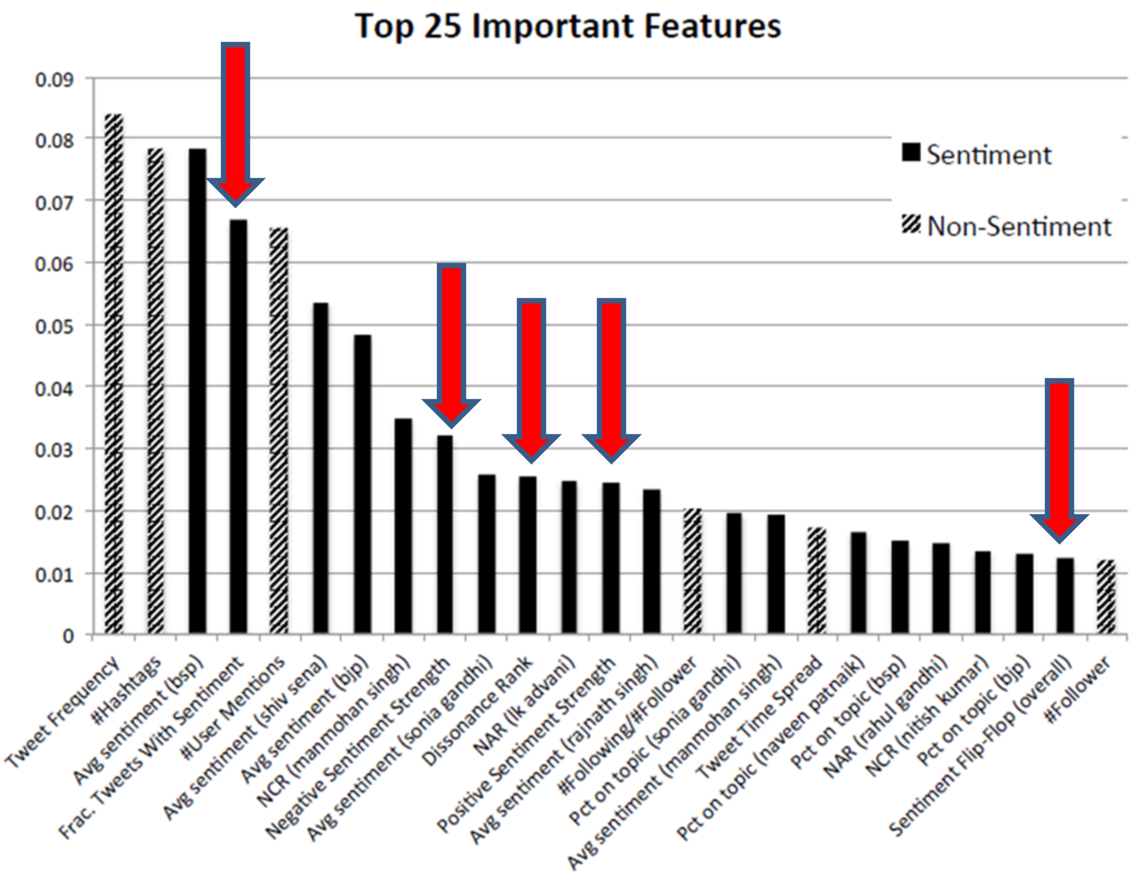
Can extend agreement rank and dissonance rank similarly

Predictive Accuracy



Which of the features do you think are the most important?

Most Important Features



19 of the 25 top features are sentiment related

THE DARPA TWITTER BOT CHALLENGE

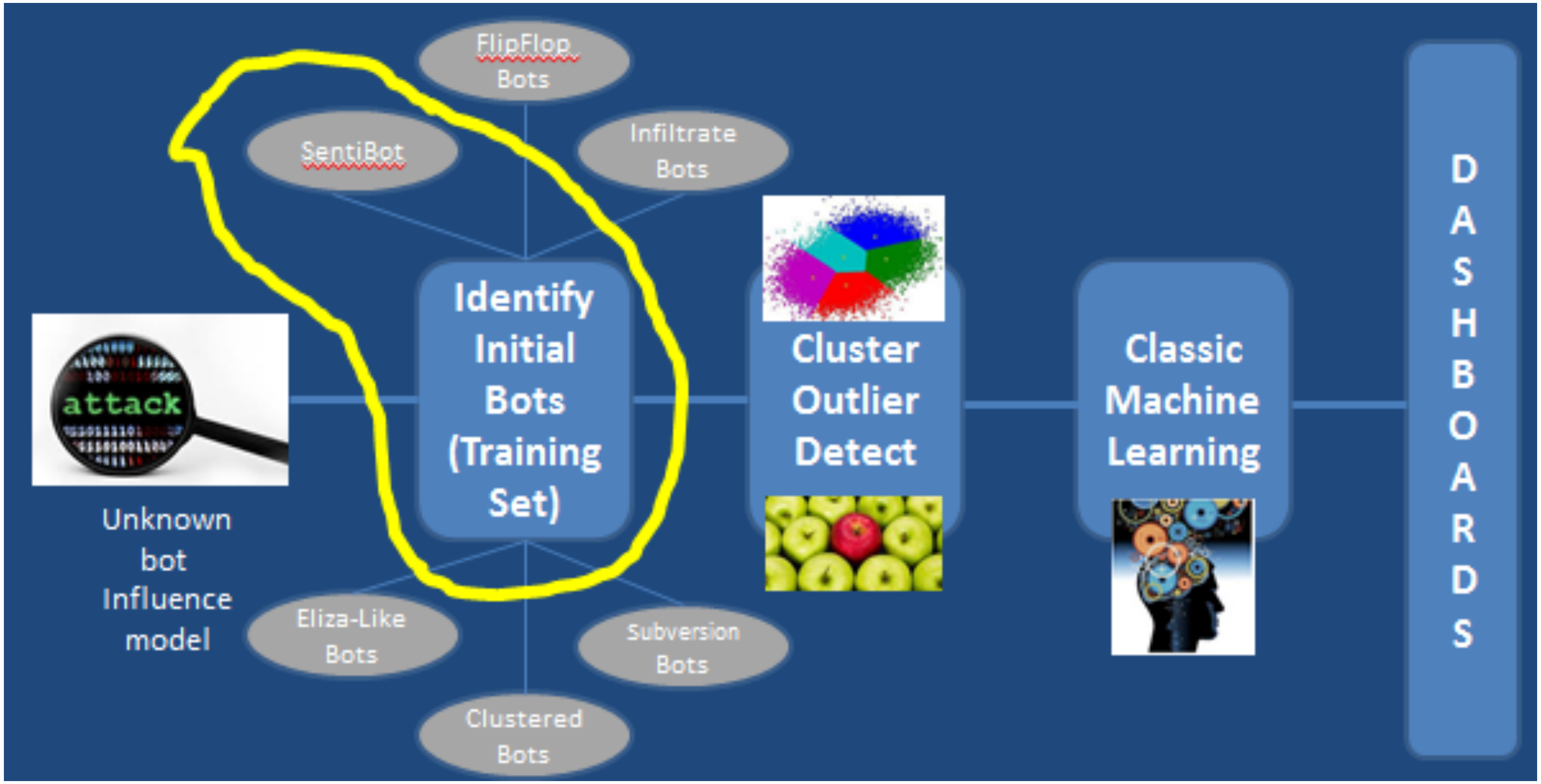
The DARPA Twitter Bot Challenge
V.S. Subrahmanian et al.
IEEE Computer, June 2016, pages 38-46

Goal: Identify all influential bots in DARPA-provided data.

Many classes of features were exploited:

- Tweet Syntax.
- Tweet Semantics (content topics and sentiment).
- Temporal Behavior Features
- User Profile Features
- Network Features.

Heterogeneity of Methods Used



Human in the loop process used to identify bots used in new social media influence campaigns including adversary strategies never seen before.

Conclusion

- Identifying bad actors varies from one type of online social source to another.
- Single paradigm for bad actor identification is elusive.
- Still can get good results in special cases.
- Tune it to your use case!

Future Directions

- Deal with dynamically evolving behavior of bad actors
- Deal with ‘smart’ bad actors
- Language agnostic algorithms
- Cross-platform detection

QUESTIONS?

Slides available at
http://bit.ly/keynote-cybersafety2016