ABSTRACT

Title of dissertation: CHARACTERIZATION AND DETECTION OF MALICIOUS BEHAVIOR ON THE WEB

Srijan Kumar, Doctor of Philosophy, 2017

Dissertation directed by: Professor V.S. Subrahmanian
Department of Computer Science

Web platforms enable unprecedented speed and ease in transmission of knowledge, and allow users to communicate and shape opinions. However, the safety, usability and reliability of these platforms is compromised by the prevalence of online malicious behavior – for example 40% of users have experienced online harassment. This is present in the form of malicious users, such as trolls, sockpuppets and vandals, and misinformation, such as hoaxes and fraudulent reviews. This thesis presents research spanning two aspects of malicious behavior: characterization of their behavioral properties, and development of algorithms and models for detecting them.

We characterize the behavior of malicious users and misinformation in terms of their activity, temporal frequency of actions, network connections to other entities, linguistic properties of how they write, and community feedback received from others. We find several striking characteristics of malicious behavior that are very distinct from those of benign behavior. For instance, we find that vandals and fraudulent reviewers are faster in their actions compared to benign editors and reviewers, respectively. Hoax articles are long pieces of plain text that are less coherent and created by more recent editors, compared to non-hoax articles. We find that sockpuppets are created that vary in their deceptiveness (i.e., whether they pretend to be different users) and their supportiveness (i.e., if they support arguments of other sockpuppets controlled by the same user).

We create a suite of feature based and graph based algorithms to efficiently detect malicious from benign behavior. We first create the first vandal early warning system that accurately predicts vandals using very few edits. Next, based on the properties of Wikipedia articles, we develop a supervised machine learning classifier to predict whether an article is a hoax, and another that predicts whether a pair of accounts belongs to the same user, both with very high accuracy. We develop a graph-based decluttering algorithm that iteratively removes suspicious edges that malicious users use to masquerade as benign users, which outperforms existing graph algorithms to detect trolls. And finally, we develop an efficient graph-based algorithm to assess the fairness of all reviewers, reliability of all ratings, and goodness of all products, simultaneously, in a rating network, and incorporate penalties for suspicious behavior.

Overall, in this thesis, we develop a suite of five models and algorithms to accurately identify and predict several distinct types of malicious behavior – namely, vandals, hoaxes, sockpuppets, trolls and fraudulent reviewers – in multiple web platforms. The analysis leading to the algorithms develops an interpretable understanding of malicious behavior on the web.

CHARACTERIZATION AND DETECTION OF MALICIOUS BEHAVIOR ON THE WEB

by

Srijan Kumar

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Advisory Committee:
Professor V.S. Subrahmanian, Chair/Advisor
Professor Ashok Agrawala
Professor John Dickerson
Professor Christos Faloutsos
Professor Jennifer Golbeck

Dedication

Dedicated to my family — I love you all.

Acknowledgements

I owe my gratitude to all the people who have made this thesis possible.

I consider myself extremely lucky to have been mentored by not one, but three incredible researchers during my PhD – VS Subrahmanian, Christos Faloutsos and Jure Leskovec.

I am very grateful to my advisor V.S. Subrahmanian for guiding me through my PhD. His encouragement and patience with me as a young graduate student, his support throughout graduate school, and his ever-enthusiastic and practical approach towards research and life have invaluably shaped my perspective and will continue to do so going forward. I want to thank VS for giving me the freedom and opportunity to work with other researchers during my PhD, and especially for the collaborations with Christos and Jure.

I am equally grateful to my two mentors – Christos Faloutsos and Jure Leskovec. My collaborations with both of them and their research groups have been the one of the most fruitful and enriching experience during my PhD. Christos' patient and persistent approach towards research is very inspiring. Thank you Christos for the several productive brainstorming sessions that were a streaming flurry of ideas. Jure's creative, dynamic and bold approach to solving practical real-world problems has gone a long way in shaping my thinking. Thanks Jure for pushing me and my research forward, and guiding me towards exciting research directions.

I would like to thank the rest of my dissertation committee – Jennifer Golbeck, John Dickerson and Ashok Agrawala – for taking out their invaluable time to give me feedback on my thesis and serving on my committee.

My research journey has been exciting and fruitful thanks to the awesome bunch of researchers with whom I had the pleasure to work with and the privilege to become friends along the way. I am obliged to Francesca Spezzano for her patient, thoughtful and honest research style that shaped my research during my early UMD days. Thanks to Bob West for closely mentoring

me during my Stanford visit, showing me the fun in research, and for advice in general. I am also grateful to the several very smart fellow students I have had the chance to work with – Justin Cheng, Vlad Niculae, Kartik Nayak, and Andrew Miller. Additionally, I have been lucky to work on interesting problems with several dynamic professors during my graduate life – Tudor Dumitras, Jordan Boyd-Graber, Cristian Danescu-Niculescu-Mizil, Elaine Shi – and in the process, learn from their unique working styles.

My graduate school journey has been made so much easier thanks to the outstanding staff at UMD. I owe special thanks to Barbara Lewis for her patient, organized, helpful, and never-say-no attitude towards any matter. I also owe my gratitude towards Jennifer Story, Arlene Shenk, Yerty Valenzuela, Fritz McCall, Tom Ventsias, Derek Yarnell and several other UMIACS staff for their consistent help and support. Special thanks to Tom for his delicious pasta! Thanks are also due to folks at CMU and Stanford for smoothly managing my visits – Marilyn Walgora, Rok Sosic, Marianne Siroker, Yesenia Gallegos – and to the two best sysadmins – Andrej Krevl and Peter Kacin.

My research adventures started way back in my undergraduate days at IIT Kharagpur. Thanks to the excellent and caring guidance of Partha P. Chakraborti, who showed me the pleasure of research. I want to thank Sudip Roy for taking me under his wings and mentoring me closely, which imbibed in me the spirit of research. I also want to thank Niloy Ganguly, Animesh Mukherjee and Tanmoy Chakraborty for introducing me to the fun research area that is Data Mining, that I eventually went to do my PhD in. I have been very influenced by the highly dedicated and passionate teaching styles of Aurobindo Gupta and Goutam Biswas – it has influenced my own way of mentoring and teaching. I learned a lot during my summer internships with Jason Corso at UB and Sameep Mehta at IBM Research. Thanks to them for motivating me, and guiding me during my visits and beyond.

I consider myself fortunate to be surrounded by so many wonderful people. Thanks to Anshul Sawant, Noseong Park, Chanhyun Kang, Francesca Spezzano, Edoardo Serra and Aaron Mannes for welcoming me into the LCCD group as a young graduate student, giving great advice during and even after leaving UMD. I also want to thank Tanmoy Chakraborty, Chiara Pulice, Joseph Barrow, Des Chandok, Sachin Grover and Liqian Zhang for the fun times spent in working and hanging out together. My stay in Maryland has been made so much fun and easy thanks to my friends – Sonali Pattnaik, Amit Kumar, Ayan Mallik, Proloy Das, Nitesh Mehta, Abhish Dev, Siddharth Santra, and so so many more. I am thankful to my friends at CMU and Stanford for making my visits fun and memorable – Hemank Lamba, Alfred Nguyen, Vivek Bagaria, Raghav Gupta, Kaushik Thakkar, Kijung Shin, Jinoh Oh, Alex Beutel, Vagelis Papalexakis, Neil Shah, Dhivya Eshwaran, Bryan Hooi, Hyun-Ah Song, Bob West, Will Hamilton, Himabindu Lakkaraju, Ashwin Paranjape, Tim Althoff, David Hallac, and David Jurgens.

Finally, I want to thank my family for their ever-lasting love, support, encouragement, and unwavering faith in me. They are the greatest source of my energy and have made me what I am today. I dedicate this thesis to them.

Thank you everyone!

Table of Contents

List of Tables x

List of Figures xi

I Introduction and Background 1

1 Introduction 2
1.1 Motivation . . . 2
1.2 Overview and contributions . . . 5
1.2.1 Predicting vandals in Wikipedia (Chapter 3) . . . 8
1.2.2 Detecting hoax articles in Wikipedia (Chapter 4) . . . 11
1.2.3 Detecting sockpuppets in online discussion communities (Chapter 5) . . . 14
1.2.4 Detecting trolls in Slashdot (Chapter 6) . . . 17
1.2.5 Predicting trustworthy users in rating platforms (Chapter 7) . . . 20
1.3 Overarching thesis statements . . . 23

2 Background and Related Work 24
2.1 Motivation of malicious behavior . . . 24
2.2 Impact of malicious behavior . . . 26
2.3 Deception and malicious behavior . . . 27
2.4 Detection Algorithms . . . 29
2.4.1 Feature Engineering . . . 29
2.4.2 Graph structure . . . 32
2.4.3 Hybrid algorithms . . . 33

II Feature Based Algorithms for Malicious Behavior Detection 35

3 Predicting Vandals in Wikipedia 36
3.1 Introduction . . . 36
3.2 The UMDWikipedia Dataset . . . 38
3.3 Vandal vs. Benign User Behaviors . . . 41

3.3.1 Similarities between Vandal and Benign User Behavior (w/o reversion features) . . . 42
3.3.2 Differences between Vandals and Benign User Behavior (w/o reversion features) . . . 43
3.3.3 Differences between Vandals and Benign User Behavior (including reversion) . . . 44
3.4 Vandal Prediction Algorithm . . . 46
3.4.1 Wikipedia Vandal Behavior (WVB) Approach . . . 46
3.4.2 Wikipedia Transition Probability Matrix (WTPM) Approach . . . 50
3.4.3 VEWS Algorithm . . . 51
3.5 Experimental Evaluation: Vandal Prediction . . . 52
3.6 Conclusions . . . 59

4 Detecting Hoax Articles in Wikipedia 62
4.1 Introduction . . . 62
4.2 Data: Wikipedia hoaxes . . . 67
4.3 Real-world impact of hoaxes . . . 68
4.3.1 Time till discovery . . . 68
4.3.2 Pageviews . . . 70
4.3.3 References from the Web . . . 72
4.4 Characteristics of successful hoaxes . . . 74
4.4.1 Appearance features . . . 75
4.4.2 Link network features . . . 77
4.4.3 Support features . . . 80
4.4.4 Editor features . . . 82
4.5 Experimental Evaluation: Automatic hoax detection . . . 83
4.5.1 Classification tasks . . . 85
4.5.2 Results . . . 88
4.6 Human guessing experiment . . . 89
4.6.1 Methodology . . . 90
4.6.1.1 Quality assurance in human guessing experiment . . . 91
4.6.2 Results . . . 91
4.7 Conclusion . . . 95

5 Detecting sockpuppets in online discussion communities 97
5.1 Introduction . . . 97
5.2 Data and Definitions . . . 100
5.3 Characterizing Sockpuppetry . . . 103
5.3.1 Do puppetmasters lead double lives? . . . 104
5.3.2 Linguistic Traits of Sockpuppets . . . 105
5.3.3 Activity and Interactions . . . 107
5.3.4 Reply network structure . . . 109
5.4 Types of Sockpuppetry . . . 110
5.4.1 Deceptiveness: Pretenders vs. non-pretenders . . . 110
5.4.2 Supporters vs. Dissenters . . . 112
5.5 Experimental Evaluation: Detecting Sockpuppets . . . 114
5.5.1 Is an account a sockpuppet? . . . 116
5.5.2 Are two accounts sockpuppet pairs? . . . 117

5.6 Discussion and Conclusion . . . 117

III Graph Based Algorithms for Malicious Behavior Detection 127

6 Detecting Trolls in Slashdot 128
6.1 Introduction . . . 128
6.2 Signed Social Networks . . . 130
6.3 Related Work . . . 133
6.3.1 Centrality Measures for SSNs . . . 133
6.3.2 Requirements of a good scoring measure . . . 137
6.3.3 Attack Models . . . 139
6.4 Decluttering Operations . . . 142
6.5 TIA Algorithm . . . 145
6.6 Experimental Evaluation: Detection Trolls . . . 148
6.6.1 Slashdot Zoo Dataset Description . . . 148
6.6.2 Experiment . . . 148
6.7 Conclusion . . . 155

7 Predicting Trustworthy Users in Rating Platforms 156
7.1 Introduction . . . 156
7.2 Related Work . . . 159
7.3 FairJudge Formulation . . . 161
7.3.1 Fairness, Goodness and Reliability . . . 161
7.3.2 Addressing Cold Start Problems . . . 166
7.3.3 Incorporating Behavioral Properties . . . 167
7.4 The FairJudge Algorithm . . . 169
7.4.1 Theoretical Guarantees of FairJudge . . . 171
7.5 Experimental Evaluation: Detecting Fraudulent Users . . . 172
7.5.1 Datasets: Rating Networks . . . 173
7.5.2 Baselines . . . 175
7.5.3 Experiment 1: Unsupervised Prediction . . . 176
7.5.4 Experiment 2: Supervised Prediction . . . 177
7.5.5 Experiment 3: Robustness of FairJudge . . . 179
7.5.6 Experiment 4: Importance of Network, Cold Start and Behavior . . . 180
7.5.7 Experiment 5: Linear scalability . . . 181
7.5.8 Discoveries . . . 181
7.6 Conclusions . . . 183

IV Concluding Remarks 184

8 Conclusion, Impact and Future Directions 185
8.1 Conclusion . . . 185
8.2 Impact . . . 188
8.3 Future Directions . . . 189
8.3.1 Malicious behavior across multiple platforms . . . 189
8.3.2 Group and coordinated malicious behavior . . . 190

8.3.3 Adversarial model of maliciousness 191
8.3.4 Unified theory of maliciousness 192

V Appendices 193

9 Proofs for the FairJudge Algorithm 194
9.1 Proof for Lemma 1: Lemma 1 195
9.2 Proof for Theorem 1: Convergence Theorem 196
9.3 Proof for Corrolary 1: Iterations till Convergence 200

Bibliography 202

List of Tables

1.1 Categorization of the presented algorithms in this thesis. 8

3.1 Features used in the edit_pair and user_log datasets to describe a consecutive edit (p_1, p_2) made by user u. 39

3.2 Table showing the accuracy and statistical values derived from the confusion matrix for the three approaches, on the entire dataset and averaged over 10 folds (without reversion features). The positive and negative class represent benign and vandal users, respectively. 53

4.1 Number of inlinks per hoax article ("SE" stands for search engines, "SN" for social networks). 74

4.2 Features used in the random-forest classifiers. 84

4.3 Classification results; for task 2, *cf.* Fig. 4.9(d). 88

4.4 Hoaxes (left) and non-hoaxes (right) that were hardest (top) and easiest (bottom) for humans to identify correctly. 95

5.1 Statistics of the nine online discussion communities. 101

5.2 74% of supporters, 70% of non-supporters and 58% of dissenters are pretenders. 114

5.3 Three sets of features were used to identify sockpuppets and sockpuppet pairs. . . 115

6.1 Table showing which axioms are satisfied by the centrality measures. Yes, No and Cond-yes mean that the axioms are satisfied, not satisfied and conditionally satisfied, respectively. Can't say means that nothing can be said in particular. 139

6.2 Table showing which centrality measure successfully prevents malicious users from using the attack models *A*-*E*. Yes and Cond-Yes means the attack is always and conditionally prevented, respectively, while No and Cond-No mean the opposite. Can't say means nothing can be said in general. . . . 141

6.3 Table comparing Average Precision (in %) using TIA algorithm with different centrality measures and decluttering operations on Slashdot network. 151

6.4 Table comparing running time (in sec.) using TIA algorithm with different centrality measures and decluttering operations on Slashdot network. 152

6.5 Number of trolls found among the lowest ranked 96 users (the number of trolls in Slasdot is 96) by using TIA algorithm with different centrality measures and decluttering operations on Slashdot network. 153

6.6 Table showing Mean Average Precision (MAP) and runtime of the five centrality measures and TIA with SEC and the top 2 decluttering operations, averaged over 50 different versions each for 95%, 90%, 85%, 80% and 75% randomly selected nodes from the Slashdot network. 154

7.1 Our algorithm FairJudge satisfies all desirable properties. 160
7.2 Five rating networks used for evaluation. 173
7.3 Unsupervised Predictions: The table shows the Average Precision values of all algorithms in unsupervised prediction of unfair and fair users across five datasets. The best algorithm in each column is colored blue and second best is gray. Overall, FairJudge performs the best or second best in 9 of the 10 cases. nc indicates 'no convergence'. 174
7.4 Supervised Predictions: 10-fold cross validation with individual predictions as features in a Random Forest classifier. Values reported are AUC. FairJudge performs the best across all datasets. nc means 'no convergence'. 177

List of Figures

1.1 (a) Vandals edit faster than benign editors. (b) VEWS outperforms ClueBot and STiki in predicting vandals. (c) Performance of VEWS improves with more edits, and addition of ClueBot and STiki features gives more accurate system. 9

1.2 (a) Roughly 1% of Wikipedia hoaxes survive for over one year without being deleted by Wikipedia administrators. (b) Hoax articles have lower clustering coefficient, which indicates towards incoherent article content. (c) Our hoax prediction model achieves high AUC for detecting hoaxes, with editor features being the most important features. 12

1.3 (a) Some sockpuppets tend to masquerade as other users by using different user names (pretenders), while others do not (non-pretenders), (b) pretender sockpuppets receive lower than non-pretenders, and (c) our prediction model accurately identifies pairs of sockpuppet accounts. 16

1.4 (a) Our decluttering algorithm performs significantly better than existing ranking algorithms in predicting trolls in Slashdot, and (b) decluttering algorithm has stable performance when nodes from the network are removed. 18

1.5 (a) Our FairJudge algorithm consistently performs the best, by having the highest AUC, in predicting fair and unfair users with varying percentage of training labels, (b) the combination of network (N), cold start treatment (C) and behavior (B) components of FairJudge algorithm gives the best performance, and, (c) FairJudge discovered a bot-net of 40 confirmed shills of one user, rating each other positively. 21

3.1 Plots showing the distribution of different properties for UMDWikipedia and edit_pair datasets. 41

3.2 Analogies and differences between benign users and vandals. 44

3.3 (a) Importance of features (w/o reversion), and (b) Percentage of vandals and benign users with particular features (w/o reversion). 50

3.4 Plot showing variation of accuracy when training on edit log of users who started editing within previous 3 months (without reversion features). The table reports the average accuracy of all three approaches. 54

3.5 Plot showing the change in accuracy by varying the training set of users who started editing Wikipedia at most n months before July 2014. The testing is done on users who started editing in July 2014. 56

3.6 Plot showing variation of accuracy with the number of first k edits. The outer plot focuses on the variation of k from 1 to 20. The inset plot shows variation of k from 1 to 500. . . 57

3.7 Plot showing the variation of accuracy for vandal detection by considering reversions made by ClueBot NG. 58

3.8 Plot showing the variation of accuracy for vandal detection by considering k^{th} REP_USER score given by STiki. . . . 59
3.9 Plot showing the variation of accuracy for vandal detection by considering article scores given by STiki. RULE: If the user makes 1 edit in first k that gets score $> t$, then the user is a vandal. . . . 60
3.10 Figure showing effect of adding STiki and ClueBot NG's features to our VEWS features. 61

4.1 Life cycle of a Wikipedia hoax article. After the article is created, it passes through a human verification process called patrol. The article survives until it is flagged as a hoax and eventually removed from Wikipedia. . . . 66
4.2 (a) Cumulative distribution function (CDF) of hoax survival time. Most hoaxes are caught very quickly. (b) Time the hoax has already survived on x-axis; probability of surviving d more days on y-axis (one curve per value of d). Dots in bottom left corner are prior probabilities of surviving for d days. . . . 69
4.3 CCDFs of (a) number of pageviews for hoaxes and non-hoaxes (14% of hoaxes get over 10 pageviews per day during their lifetime) and (b) number of active inlinks from Web. . . . 70
4.4 Longevous hoaxes are (a) viewed more over their lifetime (gray line $y = x$ plotted for orientation; not a fit) and (b) viewed less frequently per day on average (black line: linear-regression fit). . . . 71
4.5 CCDFs of appearance features; means and medians in brackets. . . . 75
4.6 Link characteristics: CCDFs (means/medians in brackets) of (a) number of wiki links, (b) wiki-link density, and (c) Web-link density. (d) Ego-network clustering coefficient as function of ego-network size (nodes of outdegree at most 10 neglected because clustering coefficient is too noisy for very small ego networks; nodes of outdegree above 40 neglected because they are very rare). . . . 78
4.7 Support features: (a) CCDF of number of mentions prior to article creation (means/medians in brackets). (b) CDF of time from first prior mention to article creation. (c) Probability of first prior mention being inserted by hoax creator or anonymous user (identified by IP address), respectively. . . . 79
4.8 Editor features: (a) CDF of time between account registration and article creation. (b) CCDF of number of edits by same user before article creation. . . . 83
4.9 (a–c) Results of forward feature selection for tasks 1, 3, 4. (d) Performance (AUC) on task 2 as function of threshold τ. . . . 87
4.10 Human bias in the guessing experiment with respect to three appearance features f. Left boxes: difference δ of suspected hoax minus suspected non-hoax. Right boxes: difference δ^* of actual hoax minus actual non-hoax. . . . 92
4.11 Comparison of easy- and hard-to-identify hoaxes with respect to three appearance features. . . . 94

5.1 AVClub.com social network. Nodes represent users and edges connect users that reply to each other. Sockpuppets (red nodes) tend to interact with other sockpuppets, and are more central in the network than ordinary users (blue nodes). . . . 119
5.2 Varying K_{min}, the minimum number of overlapping sessions between users for them to be identified as sockpuppets. For sockpuppet pairs (blue) the time between posts, and the difference in post lengths reach a minimum value at $K_{min} = 3$. . . 120

5.3 (a) Number of sockpuppet groups, *i.e.* sockpuppets belonging to the same puppet-master. (b) The second sockpuppet in a group tends to be created shortly after the first. . . . 120
5.4 Two hypotheses how similarity of sockpuppet pairs and ordinary users relates to each other. Top: Under the double life hypothesis, sockpuppet S_1 is similar to an ordinary user O, while S_2 deviates. Bottom: Alternative hypothesis is that both sockpuppet accounts are highly different from ordinary users. . . . 121
5.5 Difference in properties of sockpuppet pairs and that of sockpuppet-ordinary pairs. Sockpuppet pairs are more similar to each other in several linguistic attributes. . . . 122
5.6 (a) Histogram for the most active topic for each sockpuppet account. (b) In a sockpuppet group, the secondary sockpuppets tend to be used alongside the primary sockpuppet. . . . 123
5.7 Comparison of egonetwork of sockpuppets and similar random users in the reply network. . . . 123
5.8 The (a) display names and (b) email addresses of the sockpuppet accounts are more similar to each other compared to similar random pairs. (c) Based on the distance of display names, sockpuppets can be *pretenders* (high distance) or *non-pretenders* (low distance). . . . 124
5.9 Differences between pretenders and non-pretenders: (a) fraction of upvotes, (b) fraction of special characters in posts, (c) number of characters per sentence, (d) average sentiment, (e) usage of first person pronoun ("I"). . . . 125
5.10 (a) Based on the fraction of common discussions between sockpuppet pairs, there are two types of sockpuppets: *independent*, which rarely post on same discussion, and *sock-only*, which only post on same discussions. (b) Increase is display name distance is highly correlated with discussion use. . . . 126
5.11 Classification performance to identify (a) sockpuppets from ordinary users and (b) pairs of sockpuppet accounts (bottom). Activity features have the highest performance. . . . 126

6.1 (Left) Example of signed social network. Filled nodes are trolls, non-filled nodes are benign users. Solid (resp. dashed) edges mean positive (resp. negative) endorsements. Edges labels are the attack models used by trolls. (Right) resulting SSN after decluttering operations (a) and (d) using SEC. . 132
6.2 Decluttering operations in TIA. All nodes are marked benign. Bold (dashed) edges denote a positive (negative) relationship. . . . 144
6.3 Troll Identification Algorithm (TIA). . . . 146
6.4 TIA algorithm iterations by using Negative Rank. . . . 147

7.1 (a) The proposed algorithm, FairJudge, consistently performs the best, by having the highest AUC, in predicting fair and unfair users with varying percentage of training labels. (b) FairJudge discovered a bot-net of 40 confirmed shills of one user, rating each other positively. 158
7.2 While most ratings of fair users have high reliability, some ratings also have low reliability (green arrow). Conversely, unfair users also give some highly reliability ratings (red arrow), but most of their ratings have low reliability. . . . 163
7.3 Toy example showing products (P_1, P_2, P_3), users (U_A, U_B, U_C, U_D, U_E and U_F), and rating scores provided by the users to the products. User U_F always disagrees with the consensus, so U_F is unfair. . . . 164

7.4 This is the set of mutually recursive definitions of fairness, reliability and goodness for the proposed FairJudge algorithm. The yellow shaded part addresses the cold start problems and gray shaded part incorporates the behavioral properties. 165
7.5 Variation of AUC with percentage of training data available for supervision. FairJudge consistently performs the best across all settings, and its performance is robust to the training percentage. 178
7.6 Change in performance of FairJudge on Alpha network in (a) unsupervised and (b) supervised experiments when different components are used: network (N), cold start treatment (C) and behavioral (B). 180
7.7 (a) FairJudge scales linearly - the running time increases linearly with the number of edges. (b) Unfair users give highly negative ratings. 181
7.8 Identified unfair users by FairJudge are (a) faster in rating, and (b) give extreme ratings. . 183

Part I

Introduction and Background

Chapter 1

Introduction

1.1 Motivation

The web is a space for all, where everybody can read, publish and share information. The interconnectedness of the web enables dissemination of information, ideas and opinions to a global audience at an unprecedented speed, which has had revolutionary effects on every aspect of life, ranging from how we live to how we do business [1]. This effect can fundamentally be attributed to the change in the mode of information transmission – while the traditional means of communication were either one-to-one (e.g., postal mail, phone call) or one-to-many (e.g., newspaper, radio), the web enables many-to-many communications [1–3]. The web allows several users to simultaneously communicate, and collaboratively create and distribute content. This advance has turned every user into a 'prosumer', i.e. both a content producer and a content consumer [4].

While user interactions on the web have led to life-changing and life-saving advances [5,6], the web has also become a major breeding ground for malicious behavior [7, 8]. Such behavior manifests itself in the form of malicious users (e.g., trolls and vandals) as well as malicious information (e.g., rumors and hoaxes) [9]. Their presence is ubiquitous, for instance, 8%–10%

of social network accounts are fraudulent [10, 11], 16% of Yelp reviews are fake [12], and 3%–4% of Wikipedia editors are vandals [13]. Malicious behavior itself is nothing new, for instance, Spanish-prisoner scams date back to the 16th century [14]. Rather, what is new is the anonymity provided by the web [15] – aptly put by cartoonist Peter Steiner as, "On the Internet, nobody knows [whether] you're a dog" [16]. Online anonymity exacerbates malicious behavior, as it reduces social restrictions and inhibitions that a person may otherwise feel in face-to-face communications, a phenomenon called the 'online disinhibition effect' [17].

Online malicious behavior is in fact widespread, with research estimating that 73% of web users have witnessed online harassment and 40% have personally experienced it [18]. The impact of online malicious behavior on people's lives have been detrimental, ranging from experiencing distress [19], offline harassment [20], leading to delinquent behavior [21], and in some cases, even leading to fatalities [22]. Further, this trend is on the rise – the number of social media related crime complaints received by the FBI has more than doubled in 2015 compared to 2014, with monetary loses of over $98 million in 2015 [23].

However, there is a wide variety of malicious users and misinformation on the web, which are all fundamentally different from one another in terms of their objectives, targets, and operation. For instance, vandals on Wikipedia make unconstructive edits to Wikipedia articles, and are quite different from trolls, who harass other users. Both of them are, in turn, different from users that exploit multiple accounts (i.e., sockpuppets) to push their point of view in heated arguments. Additionally, maliciousness largely varies by platform – the objectives of malicious behavior on e-commerce platforms, Wikipedia, and social networks are all intuitively very different from one another. While maliciousness on e-commerce platforms aims to increase (or decrease) reputation of target products for higher monetary profit, vandalism of Wikipedia may have little-to-no commercial impact, and trolling in social networks in simply done for 'fun' [24]. Consequently, the

behavior of malicious entities on each of these platforms is quite distinct, and there is no uniform pattern. Nevertheless, it is extremely important to understand and detect malicious behavior on all types of web platforms, as they have a wide reach at incredibly fast speed. Therefore, the aim of this thesis is to study and detect malicious behavior across a wide array of web platforms — namely Wikipedia, Slashdot, online discussion forums like CNN, IGN, Fox News, etc., and e-commerce and rating platforms like Epinions and Amazon — and develop models to detect them accurately, in order to maintain the safety, usability and quality of the web.

Despite being an important task, several major *challenges* lie in understanding and predicting malicious behavior [25]. First, most users and activities are benign, while only a small fraction of them engage in malicious behavior, e.g., only 3%–4% of Wikipedia editors are vandals [13]. This creates an *information imbalance*, leading to much less information about malicious behavior compared to benign behavior, often resulting in poor performance of prediction algorithms. Second, often only *limited ground-truth data* on malicious behavior is available. In most cases, generating ground-truth data requires labeling of behavior by domain experts, who manually look at several instances to mark the malicious ones. This is not scalable and typically relies on proprietary information. Even so, the ground-truth label may not span all types of malicious behavior. Third, malicious behavior is often *camouflaged* to masquerade as benign behavior. Malicious users mimic the behavior, relations, likes, connections and other attributes of benign users to prevent detection. For example, they may trick benign users into endorsing them, or multiple malicious users may coordinate to improve their reputation. Taken together, the above three aspects make our task challenging, though solving them would make the social web much safer and usable, and improve the overall web experience.

To address these challenges, we adopt a data-driven approach. With the availability of large-scale real-world data, as opposed to that orchestrated in a lab-setting, these problems can

be solved at an unparalleled scale. With billions of users using the social web every day, and servers logging all of their actions, we have detailed records of their online behavior, giving an opportunity that was previously unimaginable. Making use of all of this data effectively, we develop holistic solutions to predict malicious behavior. To do so, we study the characteristics of malicious behavior and consolidate the learned insights into practically useful prediction systems.

1.2 Overview and contributions

The goal of this dissertation is to understand malicious behavior on the web, by

1. detecting patterns in the behavior of users and activities involved in malicious behavior on the web, and

2. leveraging these learned insights to develop efficient computational models to detect malicious behavior.

We develop several data-driven models to characterize and predict the two types of malicious behavior: malicious users, such as trolls, sockpuppets, vandals and fraud reviewers, and malicious information, in the form of hoaxes and fake reviews. We study the characteristics of each of the malicious behavior. We identify entity-specific behavioral characteristics, and also draw general conclusions about the common characteristics observed across these several types of entities, whenever possible. For instance, we show that malicious users are typically much faster in their actions than benign users, with no daily periodicity and often acting in quick bursts. They also tend to collude with one another to boost their own reputation. We also find that they camouflage their behavior to masquerade as benign users, and spread misinformation that is created to resemble genuine information. Altogether, we generate a highly interpretable understanding of malicious behavior on the web. This yields practical observations that teach the reader about

'suspicious' signs of maliciousness on the web.

In order to predict malicious behavior, we create feature-based and graph-based algorithms and models specifically for each of the following five different tasks. The first three are feature-based algorithms, and the last two are graph-based.

Feature-based algorithms:

First, we identify *vandals* on Wikipedia. Vandals are editors who make unconstructive edits to Wikipedia articles. We study over 33,000 editors, both vandals and benign editors, who registered on Wikipedia in over 17 months, and characterize their properties using the type of articles each editor edits, the relation between these articles, the burstiness of their edits, and the feedback by other editors. We develop a supervised machine learning model that uses these behavioral attributes of editors as features. This model outperforms state-of-the-art vandalism detection methods, and is able to accurately predict vandals after they make 2.13 edits, on average. We show that when our model is augmented with predictions from existing vandalism detection tools, the performance improves further.

Second, we study all the *hoax articles* created on Wikipedia throughout its history since 2001. We first quantify their real-world impact, in terms of their pageviews, duration of survival and presence across the web. We find that while most hoax articles are deleted quickly, some hoaxes survive for a long time and are well cited across the web. Next, we find that there are characteristic differences between hoax and non-hoax articles in terms of their article structure and content, embeddedness in the Wikipedia network, and the editor who created the article. Using these properties as features to represent articles, we create a supervised machine learning model that is accurately able to determine whether a given article is a hoax.

Third, we study *sockpuppetry* across nine online discussion forums. Taking undue advantage of anonymity, users often create multiple accounts on web platforms, called sockpuppets.

Contrary to previous belief, we find that sockpuppets may be used for both benign purposes (e.g., separate accounts for separate interests) and malicious purposes (e.g., showing inflated support for their opinions in arguments). The writing style of sockpuppets is significantly different from that of benign users. We found that pairs of sockpuppets can vary in their deceptiveness, i.e., whether they pretend to be different users, or their supportiveness, i.e., if they support arguments of other sockpuppets controlled by the same user. Based on these differences in behavior, we create a supervised model that is able to identify whether a pair of accounts belongs to the same user with very high accuracy.

Graph-based algorithms:

Fourth, we develop a novel 'decluttering' algorithm to identify *trolls* in the Slashdot social network. Trolls harass other users in the platform. The decluttering algorithm iteratively identifies and removes suspicious edges which malicious users use to masquerade as benign users. It performs significantly better, with over 3 times the accuracy compared to existing algorithms, and is 25-50 times faster as well.

Finally, we develop an algorithm to identify *fraudulent reviewers* in rating platforms, which can represented as bipartite rating graphs where *users* give rating scores to *products*. We develop an iterative algorithm that uses the rating graph to calculate the fairness of each reviewer, the reliability of each rating and the quality of each product, simultaneously. We further incorporate penalties for suspicious behavioral properties of reviewers and products. Our model automatically learns the relative importances of the rating graph and the penalties. Together this model outperforms several existing algorithms for predicting fraudulent reviewers across many rating platforms.

Overall, we develop a suite of five models and algorithms to accurately detect several distinct types of malicious behavior across various web platforms. The analysis leading to the algo-

Feature-based algorithm	Graph-based algorithm
Vandal detection (Chapter 3)	Troll detection (Chapter 6)
Hoax detection (Chapter 4)	Fraud reviewer detection (Chapter 7)
Sockpuppets detection (Chapter 5)	

Table 1.1: Categorization of the presented algorithms in this thesis.

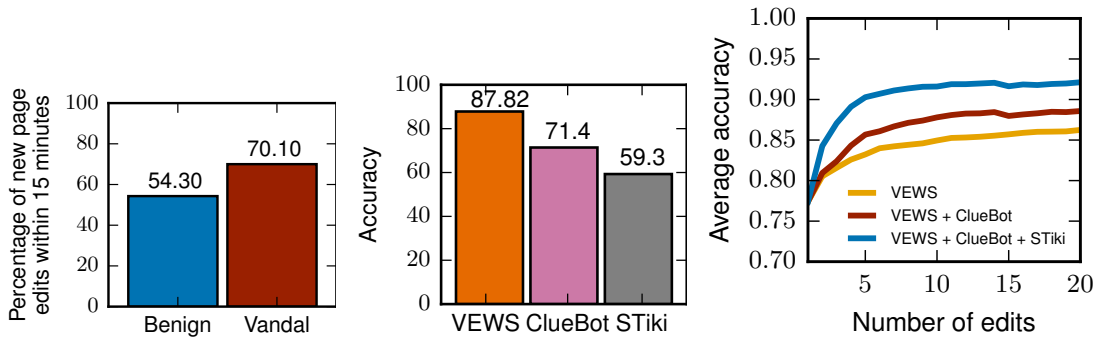
rithms develops interpretable understanding of malicious behavior on the web.

Now we give a brief explanation of each of these algorithms.

1.2.1 Predicting vandals in Wikipedia (Chapter 3)

Wikipedia is one of the largest information sources in the world, yet its quality and integrity is compromised by a relatively small number of vandals individuals who carry out acts of vandalism that Wikipedia defines as "any addition, removal, or change of content, in a deliberate attempt to compromise the integrity of Wikipedia" [26].

To address this, we create a *vandal early warning system*, called VEWS [13]. It aim to identify vandals before any human or known vandalism detection system reports vandalism so that they can be brought to the attention of Wikipedia administrators. We create a unique dataset of over 33,000 editors who registered on Wikipedia in over 17 months, and a total of over 770,000 edits. We first study the editing behavior of vandal and benign editors in Wikipedia. Each user is modelled via the articles he edits, the relation between these articles, the burstiness of his edits, and the feedback given to his edits by other editors. We find significant differences between the behavior of vandals and benign editors. For instance, Figure 1.1(a) shows that vandals make much faster edits compared to benign editors as indicated by a much larger fraction of edits made by vandals within 15 minutes of a previous edit. Further, we find that vandals make incoherent edits, by editing unrelated pages consecutively, while benign editors edit consecutive pages in similar



(a) (b) (c)

Figure 1.1: (a) Vandals edit faster than benign editors. (b) VEWS outperforms ClueBot and STiki in predicting vandals. (c) Performance of VEWS improves with more edits, and addition of ClueBot and STiki features gives more accurate system.

category or linked closely to each other.

Using these behavioral insights, we define a set of novel 'behavioral features', which is used to represent each editor. These features lie in two categories – as raw features describing his consecutive edits, and as a compressed summary of his entire edit history generated by a trained auto-encoder. The former set of features is created by deriving distinguishing features between vandals and benign editors, based on the above behavioral analysis. For the latter set of features, all the edits of a user are converted into a transition matrix, from the relation between consecutive edits. These transition matrices are compressed using an auto-encoder to reduce noise, and this compressed representation is used as the feature for the user. Both these set of features perform well in identifying vandals, with over 80% accuracy in both cases. We create a vandal early warning system, called VEWS, using both the set of features together. We show that VEWS has over 85% accuracy in identifying vandals in this dataset with a 10-fold cross-validation. Moreover, when we classify using data from the previous n months up to the current month, we get almost 90% accuracy. VEWS outperforms current leaders in vandalism detection - ClueBot NG (71.4% accuracy) and STiki (74% accuracy). Figure 1.1(b) shows the performance of the three algorithms.

Nonetheless, VEWS benefits from ClueBot NG and STiki - combining all three gives the best predictions. This is shown in Figure 1.1(c), where we see that the performance increases as more number of edits are observed for prediction, and that the performance improves when VEWS is combined with the existing systems.

Additionally, VEWS is very effective in *early identification* of vandals. VEWS detects far more vandals (15,203) than ClueBot NG (12,576). On average, VEWS predicts a vandal after it makes (on average) 2.13 edits, while ClueBot NG needs 3.78 edits. Not surprisingly, the combination of VEWS and ClueBot NG gives a fully automated system that needs no human input to detect vandals.

Overall, this algorithm makes the following contributions:

• **Characterizing vandals**: By studying over 33,000 editors, we find characteristic differences between vandals and benign editors in terms of the articles they edit, the relation between these articles, the burstiness of their edits, and the feedback given to their edits by other editors.

• **Effectiveness**: We successfully learn a supervised machine learning model that predicts vandals with over 90% accuracy, significantly outperforming existing algorithms.

• **Early warning system**: Our learned model accurately predicts vandals as early as 2.13 edits, on average, thus acting as an early warning system.

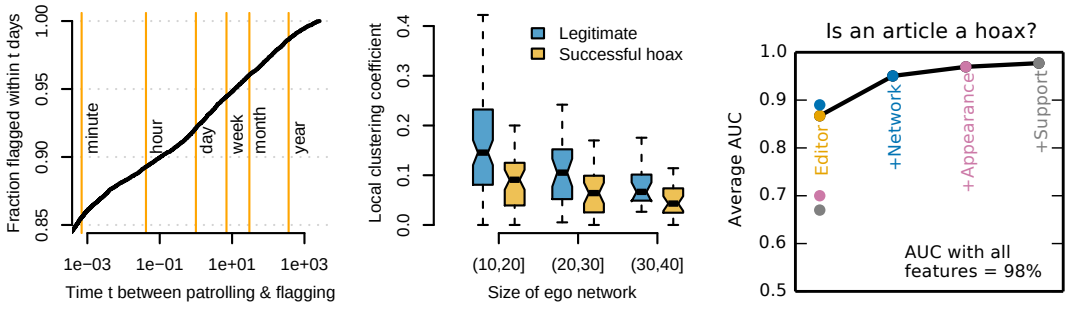
Impact.

• This research will be implemented in parts at the Wikimedia Foundation to support the English Wikipedia.

• This research has been replicated in parts in course projects at BITS Pilani, India [27] and DJ Sanghvi College of Engineering, India [28].

• This research has been invited for talk at CyberSafety workshop at CIKM 2016.

• This research has been taught in tutorials at ASONAM 2016 and WWW 2017 conferences.



(a) (b) (c)

Figure 1.2: (a) Roughly 1% of Wikipedia hoaxes survive for over one year without being deleted by Wikipedia administrators. (b) Hoax articles have lower clustering coefficient, which indicates towards incoherent article content. (c) Our hoax prediction model achieves high AUC for detecting hoaxes, with editor features being the most important features.

• This research has been taught in the following courses: University of Maryland CMSC 422 and 2017 Big Data winter school, Bari, Italy.

1.2.2 Detecting hoax articles in Wikipedia (Chapter 4)

Hoaxes on Wikipedia are completely fabricated articles that are made to masquerade as the truth [29]. These articles harm the trustworthiness of Wikipedia. We therefore study over 20,000 Wikipedia hoax articles that have been created, but later identified and deleted, throughout its existence [30].

We first quantitatively measure the *impact of Wikipedia hoaxes* by quantifying (a) how long they last, (b) how much traffic they receive, and (c) how heavily they are cited on the Web. We find that most hoaxes have negligible impact along all of these three dimensions, but a small fraction receives significant attention: roughly 1% of hoaxes are not detected for over a year after creation (Figure 1.2(a)), and 1% of hoaxes are viewed over 100 times per day on average before being uncovered.

Second, we delineate typical *characteristics of hoaxes* by comparing them to legitimate articles. We also study how successful (*i.e.*, long-lived and frequently viewed) hoaxes compare to failed ones, and why some truthful articles are mistakenly labeled as hoaxes by Wikipedia editors. In a nutshell, we find that on average successful hoaxes are nearly twice as long as legitimate articles, but that they look less like typical Wikipedia articles in terms of the templates, infoboxes, and inter-article links they contain. Further, we find that the "wiki-likeness" of legitimate articles wrongly flagged as hoaxes is even lower than that of actual hoaxes, which suggests that administrators put a lot of weight on these superficial features when assessing the veracity of an article.

The importance of the above features is intuitive, but in our analysis we find that less immediately available features are even more telling of hoax articles. For instance, we find that hoax articles are more incoherent, estimated from the clustering coefficient of its local hyperlink structure, shown in Figure 1.2(b). For an article u, the article itself, the other Wikipedia articles it connects to via hyperlinks, and any links between them forms u's ego-network. The clustering coefficient of this ego-network measures how well-connected and inter-related each of these articles are. As seen in the figure, we observe that the clustering coefficient of legitimate articles is much higher than that of hoax articles, for any number of hyperlinks, indicating that hoax creators put irrelevant and incoherent hyperlinks in the article only to look legitimate.

The creator's history of contributions made to Wikipedia before a new article is created is a further major distinguishing factor between different types of articles: most legitimate articles are added by established users with many prior edits, whereas hoaxes tend to be created by users who register specifically for that purpose.

Our third contribution consists of the application of these findings by building machine-learned classifiers for a variety of tasks revolving around hoaxes, such as deciding whether a given article is a hoax or not. We obtain good performance; *e.g.*, on a balanced dataset, where guessing

would yield an accuracy of 50%, we achieve 91%. Figure 1.2(c) shows the improvement in AUC as different set of features are added to the model for predicting hoaxes from non-hoaxes, and interestingly, editor features alone have reasonably high performance (ROC AUC = 0.89). To put our research into practice, we finally find hoaxes that have not been discovered before by running our classifier on Wikipedia's entire revision history. For example, our algorithm identified the article about "Steve Moertel", an alleged Cairo-born U.S. popcorn entrepreneur, as a hoax. The article was deleted by an editor who confirmed the articles hoax status after we had flagged it – and after it had survived in Wikipedia for 6 years and 11 months.

Finally, we assess how good humans are at telling apart hoaxes from legitimate articles in a typical reading situation, where users do not explicitly fact-check the article by using a search engine, following up on references, etc. To this end, we design and run an experiment involving human raters who are shown pairs consisting of one hoax article and one non-hoax article, and asked to decide which one is the hoax by just inspecting the articles. We find that human accuracy on this task is only 66% and is handily surpassed by our classifier, which achieves 86% on the same test set. The reason is that humans are biased to believe that well-formatted articles are legitimate and real, whereas it is easy for our classifier to see through it by also considering features computed from other articles (such as the article's ego-network clustering coefficient) as well as the creator's edit history.

Overall, this algorithm makes the following contributions:

• **Impact of hoaxes**: We quantify the impact of all 20,000 hoax articles of Wikipedia by measuring their survival time, the traffic they receive and their presence across the web. We find that while most hoaxes are caught soon, roughly 1% of hoaxes are highly impactful.

• **Characterizing hoaxes**: We study and find characteristic differences between hoax and non-hoax articles in terms of their article structure and content, embeddedness into the rest of Wikipedia,

and features of the editor who created the article.

• **Effectiveness**: We successfully apply our findings to address a series of classification tasks, most notably to determine whether a given article is a hoax, which achieves an accuracy of 91%.

Impact.

• This research has been included in the library guide of Hillsborough Community College [31].

• This research has been invited for talks at CyberSafety workshop at CIKM 2016, Wikipedia Research 2016, and VOGIN-IP conference 2017.

• This research has been taught in tutorials at ASONAM 2016, CIKM 2016, and WWW 2017 conferences.

• This research has been taught in the following courses: 2017 Big Data winter school, Bari, Italy, University of Freiburg's Web Science Seminar, University of Waterloo's CS 698 and Social and Economic Networks, and University of Alberta's CMPUT 605 courses.

1.2.3 Detecting sockpuppets in online discussion communities (Chapter 5)

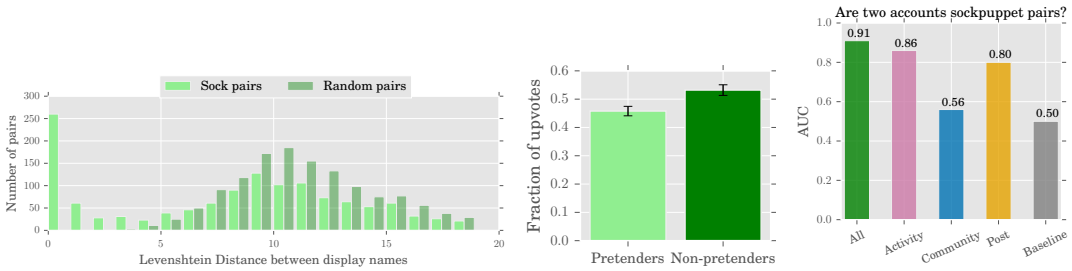
The anonymity afforded by web platforms often leads to some users deceiving others using multiple accounts, or *sockpuppets* [15], in order to manipulate public opinion [32, 33] and vandalize content (e.g., on Wikipedia [34]).

In this part, we focus on identifying, characterizing, and predicting sockpuppetry by studying *nine online discussion communities* [35]. We broadly define a *sockpuppet* as a user account that is controlled by an individual (or *puppetmaster*) who controls at least one other user account. By considering less easily manipulated behavioral traces such as IP addresses and user session data, we automatically identified 3,656 sockpuppets comprising 1,623 *sockpuppet groups*, where a group of sockpuppets is controlled by a single puppetmaster.

Studying these identified sockpuppets, we discover that sockpuppets differ from ordinary users in terms of how they write and interact with other sockpuppets. Sockpuppets have unique linguistic traits, for example, using more singular first-person pronouns (e.g., "I"), corroborating with prior work on deception [32]. They also use fewer negation words, perhaps in an attempt to appear more impartial, as well as fewer standard English parts-of-speech such as verbs and conjunctions. Suggesting that sockpuppets write worse than ordinary users on average, we find that posts are more likely to be downvoted, reported by the community, and deleted by moderators. Sockpuppets also start fewer discussions.

Examining pairs of sockpuppets controlled by the same puppetmaster, we find that sockpuppets are more likely to post at the same time and post in the same discussion than random pairs of ordinary users. By studying the network of user replies, we find that sockpuppets have a higher pagerank and higher local clustering coefficient than ordinary users, suggesting that they are more important in the network and tend to generate more communication between their neighbors. Further, we find that pairs of sockpuppets write more similarly to each other than to ordinary users, suggesting that puppetmasters tend not to have both "good" and "bad" accounts.

While prior work characterizes sockpuppetry as malicious [34, 36, 37], not all the sockpuppets we identified were malicious. Figure 1.3(a) shows that in some sockpuppet groups, sockpuppets have display names significantly different from each other, measured by their Levenshtein distance. In some other groups, sockpuppets have more similar display names. On the other hand, a pair of random users tend to have very distinct display names on average. Our findings suggest a dichotomy in how deceptive sockpuppets are – some are *pretenders*, that masquerade as separate users, while others are *non-pretenders*, that is sockpuppets that are overtly visible to other members of the community. Pretenders tend to post in the same discussions and are more likely to have their posts downvoted, reported, or deleted compared to non-pretenders (shown in Figure 1.3(b)).



(a) (b) (c)

Figure 1.3: (a) Some sockpuppets tend to masquerade as other users by using different user names (pretenders), while others do not (non-pretenders), (b) pretender sockpuppets receive lower than non-pretenders, and (c) our prediction model accurately identifies pairs of sockpuppet accounts.

In contrast, non-pretenders tend to post in separate discussions, and write posts that are longer and more readable.

Our analyses also suggest that sockpuppets may differ in their supportiveness of each other. Pairs of sockpuppets controlled by the same puppetmaster differ in whether they agree with each other in a discussion. While sockpuppets in a pair mostly remain neutral towards each other (or are *non-supporters*), 30% of the time, one sockpuppet in a pair is used to support the other (or is a *supporter*), while 10% of the time, one sockpuppet is used to attack the other (or is a *dissenter*). Studying both deceptiveness and supportiveness, we find that supporters also tend to be pretenders, but dissenters are not more likely to be pretenders, suggesting that deceptiveness is only important when sockpuppets are trying to create an illusion of public consensus.

Finally, we show how our previous observations can be used to develop models for automatically identifying sockpuppetry. We demonstrate robust performance in differentiating pairs of sockpuppets from pairs of ordinary users (ROC AUC = 0.90), as well as in the more difficult task of predicting whether an individual user account is a sockpuppet (ROC AUC=0.68). As shown in Figure 1.3(c), we discover that the strongest predictors of sockpuppetry relate to interactions between the two sockpuppet accounts, as well as the interactions between the sockpuppet and the

community.

Overall, this algorithm makes the following contributions:

• **Characterizing sockpuppets**: We study several characteristics of the posting behavior, linguistic traits, as well as social network structure of sockpuppets across nine different web discussion platforms.

• **Taxonomy of sockpuppetry**: We create a taxonomy of sockpuppet behavior across two dimensions: deceptiveness and supportiveness.

• **Effectiveness**: We perform a series of prediction tasks, notably, to identify whether a pair of accounts belongs to the same underlying user or not, and achieve an AUC of 0.90.

Impact.

• This research has won the Best Paper Honorable Mention Award at WWW 2017 conference.

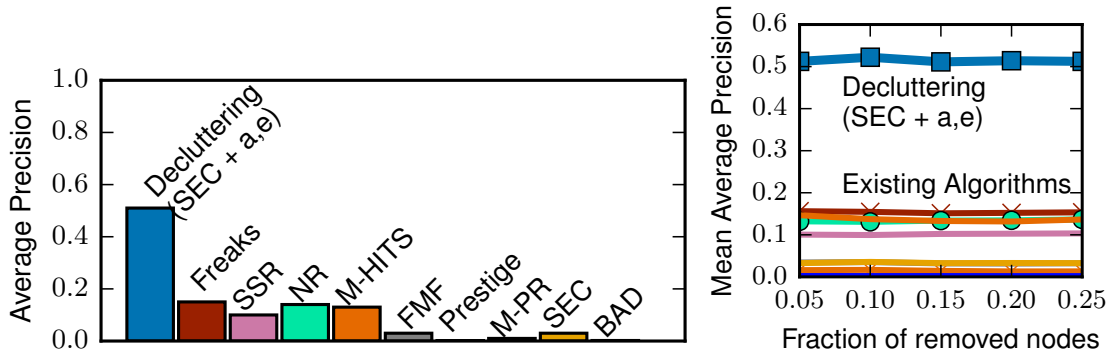
• This research has been taught at a tutorial at the WWW 2017 conference.

• This research has been covered in New Scientist, TechCrunch and WOWScience magazines.

1.2.4 Detecting trolls in Slashdot (Chapter 6)

Trolls are malicious users in social networks who post or spread misleading, offensive or nonsensical information. They are very prevalent in several social network and discussion platforms, such as Slashdot. On Slashdot, users can mark each other as friends or enemies. Trolls on this web platform tend to take a number of careful steps order to evade detection. Often, they camouflage themselves by tricking benign users into marking them as friends, and frequently collude among one another to inflate their own reputation, all the while spreading their offensive posts.

To identify these trolls, we create a *decluttering* algorithm [38] that identifies user-to-user rating edges that trolls use to hide themselves, and removes them. Intuitively, the idea is to remove some 'hay' from the 'haystack', and search for the 'needle' in the smaller 'haystack'. The



(a) (b)

Figure 1.4: (a) Our decluttering algorithm performs significantly better than existing ranking algorithms in predicting trolls in Slashdot, and (b) decluttering algorithm has stable performance when nodes from the network are removed.

removal of edges leads to a reduction in irrelevant and camouflaging edges. We propose five graph decluttering operations that reduce the network into simpler signed graph, stripped of irrelevant edges. The algorithm uses standard existing signed network centrality metrics as node ranking algorithms, to give a score to each node. The algorithm works in iterations. It first scores each node in the given Slashdot network. Using the scores and the set of decluttering operations, the algorithm then removes a set of irrelevant edges between positively scores nodes. This changes the network structure, so the ranking algorithm is used again in the reduced network. These two steps – scoring, and removing edges – are repeated iteratively till the algorithm does not identify any more edges to remove. At the end, the trolls are identified as the lowest scoring users as per the node ranking measure.

We examine several combinations of decluttering operations and several standard node ranking algorithms for signed networks. As shown in Figure 1.4(a), we find that Signed Eigenvector Centrality (SEC) and a set of two decluttering operations (operation a, which removes reciprocal positive edges between positively scored nodes, and operation c, which is the removal

of reciprocal positive and negative edge, again between positively scored nodes) achieves the best accuracy values on Slashdot. We also see that this combination of SEC with decluttering operations a and c has a stable performance, even as more nodes are removed from the network (Figure 1.4(b)). Together, we show that under appropriate settings, the decluttering algorithm has: (i) over 3 times the precision of the best node ranking algorithm to find trolls in Slashdot, (ii) retrieves over twice the number of trolls than existing algorithms, and (iii) does all this while running 25-50 times faster.

Overall, this algorithm makes the following contributions:

• **Decluttering operations**: We develop a set of five decluttering operations that can be used in conjunction with any standard node ranking algorithms.

• **Effectiveness**: We show that the decluttering algorithm significantly outperforms the ranking algorithms in identifying trolls in the Slashdot social network, achieving over 3 times the precision.

• **Speediness**: The best performing decluttering algorithm is 25-50 times faster than existing algorithms.

Impact.

• This research has been taught in tutorial at ASONAM 2016, and in the following courses: University of Maryland's CMSC 422, and 2017 Big Data winter school, Bari, Italy.

1.2.5 Predicting trustworthy users in rating platforms (Chapter 7)

Since buyers frequently look at reviews of products in e-commerce platforms before buying a product or using a vendor, there is a huge incentive for malicious entities to give fraudulent ratings to boost the overall review score of their own products and reduce the score competing products [39–42]. To identify these fraudulent reviewers, we develop an algorithm called Fair-

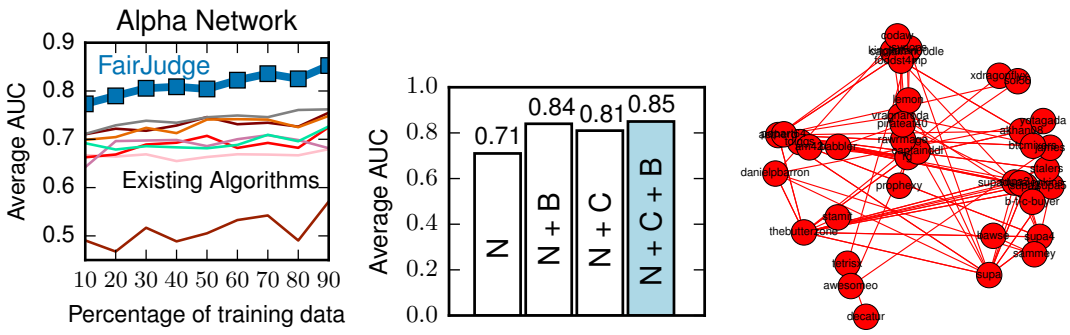
Judge.

We first develop three novel metrics to quantify the trustworthiness of users and reviews, and the quality of products, building on our prior work [43]. We model user-to-item ratings with timestamps as a bipartite graph. For instance, on an online marketplace such as Amazon, a user u rates a product p with a rating (u, p). Each user has an intrinsic level of fairness $F(u)$, each product p has an intrinsic goodness $G(p)$ (measuring its quality), and each rating (u, p) has an intrinsic reliability $R(u, p)$. Intuitively, a fair user should give ratings that are close to the goodness of the product, and good products should get highly positive reliable ratings. Clearly, these $F(u), G(p), R(u, p)$ metrics are all inter-related, and we propose three mutually-recursive equations to model them.

However, it is not possible to predict the true trustworthiness of users that have only given a few ratings. For example, users with only a few ratings, all of which are highly accurate, can be a fraudulent shill account building initial reputation [42] or it can be a genuine user. Similarly, the true quality of products that have received only a few ratings is also uncertain [44,45]. We propose a Bayesian solution to to address these *cold start problems* in our formulation by incorporating priors of users' fairness and products' goodness scores.

Additionally, the rating behavior is often very indicative of their nature. For instance, unusually rapid or regular behavior has been associated with fraudulent entities, such as fake accounts, sybils and bots [46–49]. Similarly, unusually bursty ratings received by a product may be indicative of fake reviews [50]. Therefore, we propose a Bayesian technique to incorporate users' and products' rating behavior in the formulation, by penalizing unusual behavior [48].

Combining the network, cold start treatment and behavioral properties together, we present the FairJudge formulation and an iterative algorithm to find the fairness, goodness and reliability scores of all entities together. We prove that FairJudge has linear time complexity and it is



(a) (b) (c)

Figure 1.5: (a) Our FairJudge algorithm consistently performs the best, by having the highest AUC, in predicting fair and unfair users with varying percentage of training labels, (b) the combination of network (N), cold start treatment (C) and behavior (B) components of FairJudge algorithm gives the best performance, and, (c) FairJudge discovered a bot-net of 40 confirmed shills of one user, rating each other positively.

guaranteed to converge in a bounded number of iterations.

How well does FairJudge work? We conduct extensive experiments using 5 real-world data sets – two Bitcoin user trust networks, Epinions, Amazon and Flipkart, India's biggest online marketplace. With the help of five experiments, we show that FairJudge outperforms several existing methods [44, 48, 51–56] in predicting fraudulent users. Specifically, in an unsupervised setting, FairJudge has the best or second best average precision in nine out of ten cases. Across two supervised settings, FairJudge has the highest AUC ≥ 0.85 across all five datasets. It consistently performs the best as the percentage of training data is varied between 10 and 90, as shown for the Alpha network in Figure 1.5(a). Further, we experimentally show that both the cold start treatment and behavior properties improve the performance of FairJudge algorithm, and incorporating both together performs even better (see Figure 1.5(b)).

FairJudge is practically very useful. We reported the 100 most unfair users as predicted by FairJudge in the Flipkart online marketplace. Review fraud investigators at Flipkart studied our recommendations and confirmed 80 of them were unfair, presenting a validation of the utility of

FairJudge in identifying real-world review fraud. In fact, *FairJudge is already being deployed at Flipkart.* On the Bitcoin Alpha network, using FairJudge, we discovered a botnet-like structure of 40 unfair users that rate each other positively (Figure 1.5(c)), which are confirmed shills of a single account.

Overall, this algorithm makes the following contributions:

• **Algorithm**: We propose three novel metrics called fairness, goodness and reliability to rank users, products and ratings, respectively. We propose Bayesian approaches to address cold start problems and incorporate behavioral properties. We propose the FairJudge algorithm to iteratively compute these metrics.

• **Theoretical guarantees**: We prove that FairJudge algorithm is guaranteed to converge in a bounded number of iterations, and it has *linear time complexity*.

• **Effectiveness**: We show that FairJudge outperforms nine existing algorithms in identifying fair and unfair users, conducted via five experiments on five rating networks, with $AUC \geq 0.85$.

Impact.

• The FairJudge algorithm is being used in production at Flipkart, India's largest e-commerce platform.

• This research has been taught at a tutorial at the WWW 2017 conference.

1.3 Overarching thesis statements

The following statement summarizes the thesis aptly:

> Malicious behavior differs from benign behavior both with respect to their attributes and connectivity. These differences can be used to create efficient computational models to detect them.

Chapter 2

Background and Related Work

Characterization and detection of malicious behavior is an important topic that has been studied from various different perspectives. It builds on a rich vein of prior work in motivation of malicious behavior (Section 2.1), deception in malicious behavior (Section 2.3) and finally, algorithmically, for detection (Section 2.4). This chapter reviews the relevant related research in all these areas.

2.1 Motivation of malicious behavior

The reasons for malicious behavior can be various. While on the one hand, it has been suggested that malicious behavior is done only for fun [24], on the other hand, it is often used to cheat and increase profit [57–60]. The primary motivation of malicious behavior is often money – the use of fraud reviews to boost the perceived appearance of products on e-commerce platforms leads to significant increase in profit [39–41]. There are documented cases where people have created sockpuppet accounts to give glowing reviews to their own books [61]. Malicious behavior in the form of sockpuppetry is often used to avoid being banned, to create false consensus [32,33] and support a person or a position [60], or vandalize content (*e.g.* on Wikipedia [34]).

Malicious behavior is triggered by three simple factors: presence of malicious intent, availability of suitable targets, and absence of effective checks [62]. The wide connectivity and popularity of social web platforms gives countless targets, and even though there may be malicious behavior prevention systems, malicious users may be able to circumvent them to act maliciously. Web-based malicious behavior is further exacerbated by anonymity, as web interactions reduces social restrictions and inhibitions that a person may otherwise feel in face-to-face communications [17, 62].

Malicious behavior is influenced both by individual traits and experiences. Sadism has been found to be highly correlated with trolling behavior [24]. Frequent users are more likely to be malicious than infrequent users, young users more than old, and competent users more than non-competent [60]. It has also been found social and community factors play an important role in perpetrating malicious behavior [63]. Additionally, malicious behavior may not be premeditated, but rather, encouraged by personal and community factors – negative mood of the person and noticing malicious behavior from others act as triggers for embracing malicious behavior themselves [64].

Our contributions: Our work adds to the understanding of motivation for malicious behavior in the form of sockpuppetry. While prior work has relied on small scale user surveys to understand malicious behavior motivation, we use a data-driven approach, and analyse roughly 2.8 million users and 62 million posts to understand the motivation behind sockpuppetry. We find that sockpuppets may be created for malicious as well as benign purpose – sockpuppets in discussion communities sometimes support each other, in order to portray higher support for themselves and for their point of view. Beyond malicious uses, some uses of sockpuppetry may be benign, e.g., a user may simply use different accounts to post in different topics. Sometimes, a sockpuppet may even pretend to disagree with another to gain the trust of users opposing the other one's arguments.

Based on how they comment on one another's posts, we present a categorization of sockpuppets as supporters, non-supporters and dissenters. Similarly, we find that malicious accounts in trust networks, such as Slashdot and Bitcoin-OTC, are often created to positively cross-rate each other, in order to boost their public ratings.

2.2 Impact of malicious behavior

Online malicious behavior is widespread and has far-reaching impact. Researchers have estimated that 73% of web users have witnessed online harassment and 40% have personally experienced it [18]. Their real-world impact on people's lives is detrimental, ranging from experiencing distress [19], offline harassment [20], leading to delinquent behavior [21], and in some cases, even leading to fatalities [22], and this trend is on the rise [23]. Further, on e-commerce platforms, fake reviews in e-commerce platforms lead to undue increase the real-world profit for sellers [39–41].

False information spreads widely and quickly across the web. When false information in the form of rumors, urban legends, and conspiracy theories appears in a social network, users are often led to share and disseminate it [65, 66]. There is a rich line of empirical investigations and case-based studies of how this propagation happens, *e.g.*, in Facebook [66, 67], Twitter [68], and Sina Weibo [69]. This is harmful, as having heard the rumor previously leads people to believe it [70]. Further, malicious behavior can be contagious – witnessing existing malicious behavior can encourage other users to do malicious acts themselves [64].

Our contributions: In our work, we add to this line of work in two important aspects: we quantify the impact of malicious behavior, and identify what leads people to fall victims to malicious behavior. First, we quantify the impact of Wikipedia hoaxes in terms of how long they last, how many pageviews they get, and how much they spread on the web (chapter 4). We find

that while most hoaxes are caught soon and get few pageviews, a small fraction of hoaxes survive for a long time without being detected, get hundreds of page views per day, and are well-cited from across the web, making these very impactful. We also find that vandals on Wikipedia tend to make 9.43 edits, on average, before they are caught and banned, and therefore, highly impactful in potentially harming the credibility of Wikipedia. Moreover, sockpuppets in discussion forums tend to write worse than non-sockpuppets, e.g., they make abusive posts, potentially leading to stress in the recipients of their posts. Second, we conduct experiments to understand the reasons that lead people to believe in hoaxes in Wikipedia. We show that people do not perform particularly well when trying to distinguish false information from the truth. Casual readers are only able to detect 66% of hoaxes, when presented with the task of identifying hoax from a pair of hoax and non-hoax article, which is only slightly better than random (50%). Moreover, we find that well-crafted hoaxes, i.e., the ones that look more like genuine Wikipedia articles, tend to fool humans.

2.3 Deception and malicious behavior

Malicious behavior is closely related to deception. Trolls, vandals, fraud reviewers, bots, sockpuppets and other malicious users often deceptively camouflage themselves in order to evade detection [42, 71]. Deception online is aided by the virtue of anonymity on the web [15]. Prior research suggests that the behavior of people changes when they deceive, for example, they reduce communication [72] and change the focus of their presentation [73]. When writing deceptively to hide their identity, authors tend to increase use of particles and personal pronouns, write shorter sentences, and show nervousness [32, 74–76].

Malicious users are frequently deceptive in their actions and text, for e.g., in writing reviews or giving ratings to products, or giving positive or negative edges to other users in trust networks. Being deceptive, they are able to trick benign users into endorsing them, other malicious users,

and/or products. For instance, in social networks, a benign user may endorse a malicious user after being endorsed first, not necessarily because they may actually like the malicious user. Or in an e-commerce platform, a benign user may be influenced into endorsing a bad product, when several malicious users endorse the bad product as well. Deceptive attack models have been suggested in literature to model the types of deceptiveness [77], which include: (a) *malicious collectives*, where malicious users endorse other malicious users, (b) *camouflage behind good transactions*, where malicious users can cheat some benign users to endorsing for them, (c) *malicious spies*, where malicious users make benign users to endorse for them, and themselves endorse only other malicious users, and finally (d) *camouflage behind judgements*, where malicious users attack, oppose or criticize benign users.

Our contribution: Our work adds to this line of research by finding evidence of deceptive writing styles and actions across various types of malicious users. We find that troll users in Slashdot are able to receive positive edges from non-trolls by deceiving them. Next, Some hoax articles on Wikipedia are created deceptively, and are able to fool Wikipedia volunteers into believing the hoax article is actually genuine. Such articles are well written, with several web and Wikipedia references, images, section-headings, and other attributes to look like a genuine Wikipedia article. Third, in e-commerce platforms, we find that while fraudulent users usually give many fake ratings, they also camouflage themselves by giving correct ratings to products. Finally, we find that sockpuppets vary in their deceptiveness, i.e., whether they pretend to be different users. While pretender sockpuppets use strikingly different usernames to pretend to be different people, non-pretenders use similar usernames and are not deceptive. Therefore, we identify several behavioral characteristics of deceptive behavior in malicious users and activities in this work.

2.4 Detection Algorithms

Several algorithms have been developed in prior literature to identify and predict malicious behavior. Algorithms may use the graph structure of the network or actions, associated metadata (e.g., time of actions), input from humans, or a combination of these. While some algorithms identify individual malicious users, some others identify group (or coordinated) malicious behavior. Here, we categorize the works into feature engineering based, graph structure based, and finally, a hybrid combination of these.

2.4.1 Feature Engineering

Feature engineering is one of the most widely used technique in classification tasks, and are used to model behavior. By representing each user (or action) as a point in a multi-dimentional feature space, these techniques aim to create separate regions where malicious and benign users (or actions) lie. Standard machine learning classifiers are then used to distinguish between the two.

Some of the most commonly used categories of features and their examples are listed below:

- *Activity features*, e.g., number of actions, fraction of unique actions, distribution of types of actions and its entropy.
- *Temporal features*, e.g., mean and median time between consecutive actions, total lifetime of all activities, entropy of inter-action time
- *Linguistic features*, e.g., length of text, fraction of non-character, all-capital letters, sentiment, LIWC features
- *Network structure*, e.g., degree, reciprocity, local clustering coefficient, density, centrality

value, local consensus

- *Community feedback*, e.g., number of total responses, fraction of negative response (e.g., downvotes, reports, blocks, revertions)

These five categories of features are very intuitive and effective. These features are often used in combination with each other, and have been shown to outperform their effectiveness alone in several cases [26, 45, 48, 78].

Activity features encapsulate all the actions done by the user, such as the number of such actions they make (e.g., it has been shown that malicious users perform fewer actions [13, 78, 79]), and how often they perform similar actions (e.g., malicious users are more repetitive, while benign users perform a wide range of actions [71]). These features have been shown to be effective in distinguishing malicious users from benign ones [13, 35, 78–80].

The second set of commonly used features is based on time. These features capture the temporal relationship between two consecutive actions. The intuition behind this set of features is that malicious users are active for a short period of time, and during this time period, they perform several actions. This results in lower time between consecutive posts, and smaller lifetime of actions. Certain malicious users may be more regular than benign users. For example, bots may be pre-programmed to perform certain actions every 'n' seconds [81]. This leads to lower entropy in their inter-action time. Temporal features have been shown to be very effective in identifying fake reviews [45, 50, 82], trolls [78], bots [80], and fraudulent users [48].

Next, linguistic features incorporate how users write, whenever available. Several platforms, such as review platforms like Amazon or discussion platforms, rely on textual review to express opinion and convey information. Malicious users write differently than benign users – they write shorter text [83], and are more aggressive and abusive [26, 78, 84]. They are also more opinionated [55, 80]. As these attributes are distinguishing, they have been shown to be very

effective in identifying fake reviews [55, 83], trolls [78] and vandalism edits on Wikipedia [26].

Network structure measures how the user connects to other users in user-to-user social or communication network. The connections may be attributed, for example, with sentiment. Features such as density and clustering coefficient of the local neighborhood measure how well connected it is, while features such as degree measures how many connections the user has (incoming, outgoing or total), and reciprocity measures how often other users reciprocate the connections. Consensus features measure similarity between the connections made by a user and the connections made by other users, connecting to the same user or product [44, 55]. For example, if frequently rates products with ratings that do not concur with the other ratings given to the product, then the consensus of the user is low. Network features have been shown to be very telling of malicious behavior – in some cases, the local neighborhood of the malicious users is too sparse or too dense [85–87]; malicious users may create too many connections [38], but may not be reciprocated as much [87, 88]. This set of features has effectively been used to detect various types of malicious users [38, 86, 89, 90].

Finally, community feedback measures how the other users react to the actions made by a user. Malicious users are more likely to receive negative feedback in response to their actions [13, 78]. This can be in terms of downvotes to the user's comments, reversions of the user's actions, or reporting the user or action to administrators. This set of features emerges after the user has committed his actions, and it has been identified by other users to be inappropriate or malicious. On the other hand, benign users are more likely to receive positive feedback, such as upvotes and positive comments [13, 78]. As these features directly incorporate positive or negative feedback, this set of features is very useful in identifying malicious users, as in case of detecting trolls [78], vandals [13], and sockpuppets [35].

Our contributions: Our research contributes significantly towards this line of research, by

developing feature based algorithms to detect vandals in Wikipedia (see Chapter 3), hoaxes on Wikipedia (see Chapter 4), and sockpuppets in online discussion communities (see Chapter 5). These three algorithms use several of these five category of features, wherever applicable, and achieves highly accurate classification accuracy. We find that some categories may be more important than others for prediction – activity features were more important to detect sockpuppets and hoaxes than community and temporal features. Moreover, the importances of these categories vary across different tasks. For example, community feedback in Wikipedia (in terms of reversions) is important to identify vandals, but not as important to detect sockpuppets.

2.4.2 Graph structure

Several algorithms use the local graph structure and/or the global graph structure as the primary basis of the algorithm. Instead of converting the structure into features, these methods create ranking algorithms by clustering, community detection or belief propagation.

Much work on measuring reputation propagates trustworthiness of users on social networks [91, 92]. [93] studied the propagation of trust and distrust in social network for the purpose of edge sign prediction. FraudEagle [51] is a belief propagation model to rank users, which assumes fraudsters rate good products poorly and bad products positively, and vice-versa for honest users. Starting with an initial 'belief' of each node's maliciousness, it propagates these beliefs throughout the network, till the beliefs converge. The final belief scores are used to rank nodes. Similar to trust propagation, random-walk based algorithms have been developed to detect trolls [94] and link farming from collusion on Twitter [87].

Some algorithms work on weighted networks, i.e., networks where edges have weights. These weights may be signed (positive or negative). Trust and rating networks are examples of this type of networks. [52, 54, 95] develop iterative algorithms that jointly assign scores in the

rating networks based on consensus of ratings - [54] scores each user, review and product, and [52] scores each user and product. Trustiness [54] creates a user-review-product graph and assigns a trustiness score to each user, a honesty score to reviews and reliability scores to products. Bias and Deserve [52] assigns a bias score to each user and a deserve score to each product. Both these algorithms calculate their respective scores jointly based on consensus of ratings among users.

Coordinated or group spam behavior has also been studied using graph structure, such as synchronized and lockstep behavior [86, 89, 90], and coordinated spam in rating platforms [96].

A *survey* on network-based fraud detection can be found in [97].

Our contributions: This thesis adds to this line of work by developing two graph based algorithms: decluttering (see Chapter 6) and FairJudge (see Chapter 7). Decluttering algorithm identifies suspicious edges in the network that malicious users use to masquerade themselves. These edges are removed from the network, and nodes are ranked. Suspicious edges are re-identified in this reduced network, and removed. This iterative algorithm continues till there are no more suspicious edges to be removed. Nodes are ranked in the final network.

FairJudge works on rating networks, and jointly assigns the fairness score to each user, reliability score to each edge, and goodness score to each product. The intuition behind this is that fair users give reliable ratings that are close in score to the goodness score of products. It randomly initializes these values, and then iteratively propagates these score through the network.

Both these algorithms perform significantly better than the state of the art in their respective tasks.

2.4.3 Hybrid algorithms

A combination of several of the two major genre of algorithms – feature-based and graph-based – has frequently been shown to outperform them individually. For example, SpEagle al-

gorithm [53] extends FraudEagle [51] to incorporate behavior features, and performs better in identifying fake reviews. Some techniques also incorporate human input into their algorithm. For instance, to identify vandalism on Wikipedia, combining features (e.g., NLP, reputation and metadata) from [98,99] and human input from STiki tool [100], it is possible to obtain a classifier with better accuracy [101].

Our contributions: In our research, we combine both feature-based algorithm and graph-based algorithm to detect trustworthy reviewers in rating platforms (see Chapter 7). The FairJudge algorithm combines its novel graph-based algorithm, that assigns fairness, reliability and goodness scores, with features scores derived from each user and product's behavior. The features are incorporated as BIRDNEST scores [48] in a novel way, and together, the complete FairJudge algorithm outperforms the two components individually. The FairJudge algorithm is already under deployment at Flipkart, India's largest e-commerce platform.

Further, in detecting vandals in Wikipedia (see Chapter 3), we show that merging algorithmic predictions and human input together improves the prediction. The combination of VEWS and ClueBot NG is entirely algorithmic, and when incorporated with STiki scores, that have human input, it is able perform better in predicting the vandals.

Part II

Feature Based Algorithms for Malicious Behavior Detection

Chapter 3

Predicting Vandals in Wikipedia

With over 4.6M articles, 34M pages, 23M users, and 134K active users, the English Wikipedia is one of the world's biggest information sources, disseminating information on virtually every topic on earth. However, there are two major problems that undermine the credibility of information on Wikipedia. First, vandals make unconstructive edits on Wikipedia articles, and second, hoax articles are fabricated false articles created on Wikipedia to masquerade as truth by hoaxsters. We study these two aspects of malicious behavior on Wikipedia in the next two chapters. In this chapter, we characterize the behavior of vandals, and then develop an early warning system to detect them. Later, in Chapter 4, we measure the impact of hoaxes, characterize their properties, and finally create an algorithm to detect them.

3.1 Introduction

Wikipedia is compromised by a relatively small number of vandals — individuals who carry out acts of vandalism that Wikipedia defines as "*any addition, removal, or change of content, in a deliberate attempt to compromise the integrity of Wikipedia*" [26]. Vandalism is not limited to Wikipedia itself, but is widespread in most social networks. Instances of vandalism have been

reported in Facebook (vandalism of Martin Luther King, Jr.'s fan page in Jan 2011), WikiMapia and OpenStreetMaps [102].

There has been considerable work on identifying vandalized pages in Wikipedia. For instance, ClueBot NG [103], STiki [104], and Snuggle [105] use heuristic rules and machine learning algorithms to flag acts of vandalism. There is also linguistic work on finding suspicious edits by analyzing edit content [98–101, 106]. Most of these works use linguistic features to detect vandalism.

Our goal in this chapter is the *early identification* of vandals, before any human or known vandalism detection system reports vandalism, so that they can be brought to the attention of Wikipedia administrators. This goes hand-in-hand with human reporting of vandals.

Contributions. This chapter contains five main contributions.

1. We define a novel set of "behavioral features" that capture edit behavior of Wikipedia users.

2. We conduct a study showing the differences in behavioral features for vandals vs. benign users.

3. We propose three sets of features that use no human or known vandal detection system's reports of vandalism to predict which users are vandals and which ones are benign. These approaches use the behavioral features from above and have over 85% accuracy. Moreover, when we do a classification using data from previous n months up to the current month, we get almost 90% accuracy. We show that our VEWS algorithm handily beats today's leaders in vandalism detection - ClueBot NG (71.4% accuracy) and STiki (74% accuracy). Nonetheless, VEWS benefits from ClueBot NG and STiki - combining all three gives the best predictions.

4. VEWS is very effective in early identification of vandals. VEWS detects far more vandals (15,203) than ClueBot NG (12,576). On average, VEWS predicts a vandal after it makes

(on average) 2.13 edits, while ClueBot NG needs 3.78 edits. Overall, the combination of VEWS and ClueBot NG gives a fully automated system without any human input to detect vandals (STiki has human input, so it is not fully automated).

5. We develop the unique UMDWikipedia data set that consists of about 33K users, about half of whom are on a white list, and half of whom are on a black list.

3.2 The UMDWikipedia Dataset

We now describe the UMDWikipedia dataset[1] which captures various aspects of the edits made by both vandals and benign users.[2] The UMDWikipedia dataset consists of the following components.

Black list DB. This consists of all 17,027 users that registered and were blocked by Wikipedia administrators for vandalism between January 01, 2013 and July 31, 2014. We refer to these users as *vandals*.

White list DB. This is a randomly selected list of 16,549 (benign) users who registered between January 01, 2013 and July 31, 2014 and who are not in the black list.

Edit Meta-data DB. This database is constructed using the Wikipedia API [107] and has the schema

$$(User, Page, Title, Time, Categories, M)$$

A record of the form (u, p, t, t', C, m) says that at time t', user u edited the page p (which is of type m where m is either a normal page or a meta-page[3]), which has title t and has list C of

[1]The data set is available at http://www.cs.umd.edu/~vs/vews

[2]We only studied users with registered user names.

[3]Wikipedia pages can either be normal article pages or can be discussion or "talk" pages where users may talk to each other and discuss edits.

Whether p_2 is a meta-page or normal page.
Time difference between the two edits: less than 3 minutes (very fast edit), less than 15 minutes (fast edit), more than 15 minutes (slow edit).
Whether or not p_2 is the first page ever edited by the user.
Whether or not p_2 is a page that has already been edited by the user before (p_2 is a re-edit) and, if yes - Whether or not p_1 is equal to p_2 (i.e. were two consecutive edits by the same user applied to the same page); - Whether of not a previous edit of p_2 by the user u has been reverted by any other Wikipedia user. Otherwise, p_2 is a page edited for the first time by user u. In this case, we include the following data: - the minimum number of links from p_1 and p_2 in the Wikipedia hyper-link graph: more than 3 hops, at most 3 hops, or not reachable; - the number of categories p_1 and p_2 have in common: none, at least one, or $null$ if category information is not available.

Table 3.1: Features used in the edit_pair and user_log datasets to describe a consecutive edit (p_1, p_2) made by user u.

Wikipedia categories attached to it.[4] All in all, we have 770,040 edits: 160,651 made by vandals and 609,389 made by benign users.

Edited article hop DB. This database specifies, for each pair (p_1, p_2) of pages that a user consecutively edited, the minimum distance in the Wikipedia hyper-link graph[5] from p_1 to p_2. We used the code provided by [108].

Revert DB. Just for the one experiment we do at the very end, we use the edit reversion dataset provided by [109] which marks an edit "reverted" if it has been reverted within next 15 edits on the page. [110] suggests that 94% of the reverts are detected by the method used to create

[4]Note that Wikipedia assigns a category to each article from a category tree — this therefore labels each page with the set of categories to which it belongs.

[5]This is the graph whose vertices are pages and where there is an edge from page p_1 to p_2 if p_1 contains a hyper-link to p_2.

the dataset. Therefore, we use this dataset as ground truth to know whether the edit was reverted or not. *Note that this information is not needed as a feature in our algorithm for prediction, but to analyze the property of reversion across vandals and benign users.* Observe that Revert DB also contains the information whether the reversion has been made by ClueBot NG. We use this to compare with ClueBot NG.

STiki DB. We use the STiki API [111] to collect STiki vandalism scores, and the raw features used to derive these scores (including the user reputation score). We use vandalism and user scores *only* to compare with STiki.

Edit Pair and User Log Datasets To analyze the properties of edits made by vandals and benign users, we create two additional datasets using the data in the UMDWikipedia dataset.

Edit Pair Dataset. The edit_pair dataset contains a row for each edit (u, p_1, p_2, t), where u is a user id, (p_1, p_2) is a pair of Wikipedia pages that are consecutively edited by user u, and t is the time stamp of the edit made on p_2. Note that p_1 and p_2 could be the same if the user makes two edits, one after another, on the same page. Each row contains the values of the features shown in Table 3.1 computed for the edit (u, p_1, p_2, t). These features describe the properties of page p_2 with respect to page p_1.

User Log Dataset. The chronological sequence of each consecutive pair (p_1, p_2) of pages edited by the same user u corresponds to a row in this dataset. Each pair (p_1, p_2) is described by using the features from Table 3.1. This dataset is derived from the edit_pair dataset. It captures a host of temporal information about each user, suggesting how he/she navigated through Wikipedia and the speed with which this was done.

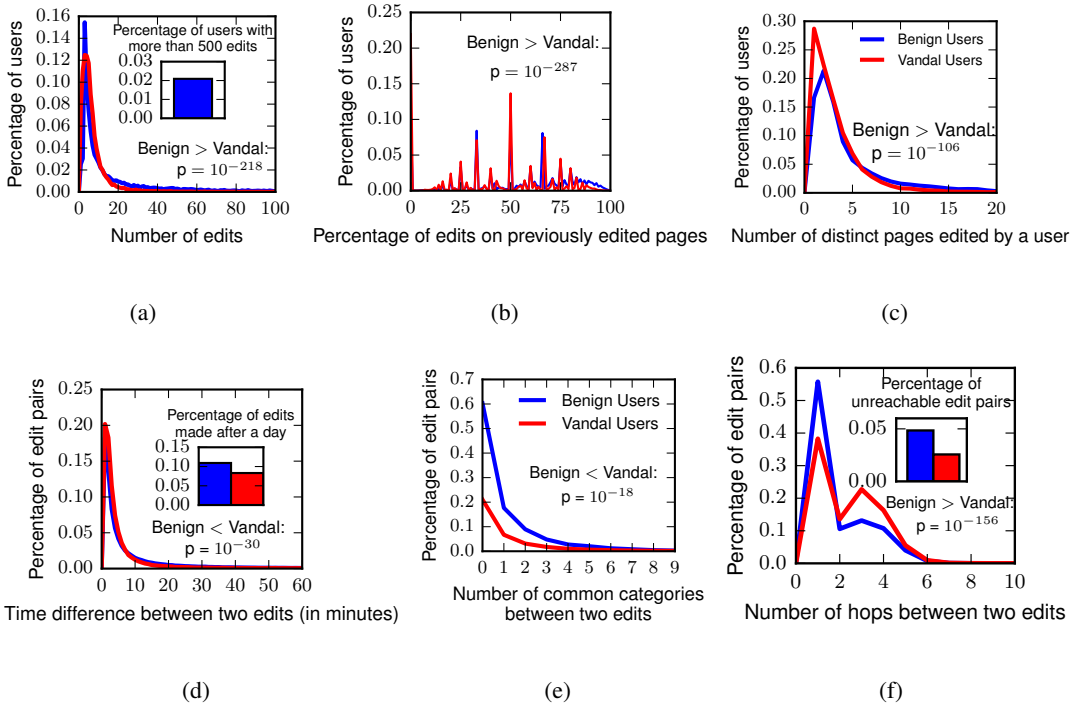


Figure 3.1: Plots showing the distribution of different properties for UMDWikipedia and edit_pair datasets.

3.3 Vandal vs. Benign User Behaviors

In this section, we statistically analyze editing behaviors of vandals and benign users in order to identify behavioral similarities and differences.

Figure 3.1 shows the distributions of different properties that are observed in the edit_pair dataset. Figures 3.1(a)- 3.1(c) show the percentage of users on the y-axis as we vary the number of edits, number of distinct pages edited and the percentage of re-edits on the x-axis. These three graphs show near identical behavior.

Figures 3.1(d)- 3.1(f) show the percentage of edit pairs (u, p_1, p_2) on the y-axis as we vary time between edits, number of common categories between edited pages p_1 and p_2 and number of hops between p_1 and p_2. The behavior of users in terms of time taken between edits is nearly identical. The last two graphs show somewhat different behaviors between vandals and benign

users. Figure 3.1(e) shows that the percentage of edit pairs involving just one, two, or three common categories is 2-3 times higher for benign users than for vandals. Likewise, figure 3.1(f) shows that for benign users, the percentage of edit pairs involving exactly one hop is 1.5 times that of vandals, but the percentage of edit pairs involving 3-4 hops is much higher for vandals than for benign users.

In all the histograms in Figure 3.1, the null hypothesis that the distribution for vandals and benign users have identical average has p-value ¿ 0.05. As this fails to say that their behavior is not similar, we do a more in-depth analysis to distinguish between them. So, we perform a frequent itemset mining step on the edit_pair and user_log datasets. Figure 3.2 summarizes the results.

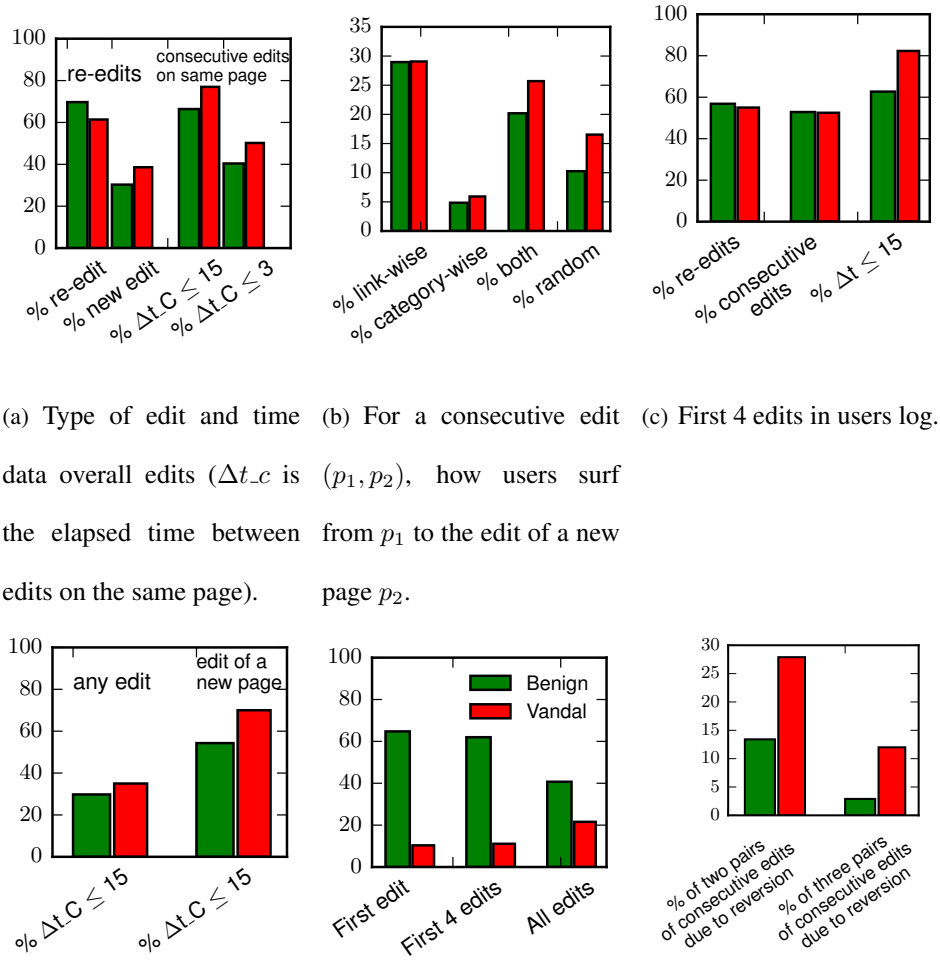
3.3.1 Similarities between Vandal and Benign User Behavior (w/o reversion features)

Figures 3.2(a), 3.2(b), and 3.2(c) show similarities between vandal and benign user behaviors.

• *Both vandals and benign users are much more likely to re-edit a page compared to editing a new page.* We see from Figure 3.2(a) that for vandals, the likelihood of a re-edit is 61.4% compared to a new edit (38.6%). Likewise, for benign users, the likelihood of a re-edit is 69.71% compared to a new edit (30.3%).

• *Both vandals and benign users consecutively edit the same page quickly.* The two rightmost bars in Figure 3.2(a) show that both vandals and benign users edit the same page fast. 77% of such edit pairs (for vandals) occur within 15 minutes – this number is 66.4% for benign users. In fact, over 50% of these edits occur within 3 minutes for vandals - the corresponding number for benign users is just over 40%.

•*Both vandals and benign users exhibit similar navigation patterns.* 29% of successively edited pages (for both vandals and benign users) are by following links only (no common cate-



(a) Type of edit and time data overall edits (Δt_c is the elapsed time between edits on the same page).

(b) For a consecutive edit (p_1, p_2), how users surf from p_1 to the edit of a new page p_2.

(c) First 4 edits in users log.

(d) Edit time.

(e) Meta-pages editing.

(f) Edit war (involves reversion).

Figure 3.2: Analogies and differences between benign users and vandals.

gory and reachable by hyperlinks), about 5% due to commonality in categories only between the successively edited pages (at least one common category and not reachable by hyperlinks), and 20-25% with commonality in both properties and linked. This is shown in Figure 3.2(b).

- *In their first few edits, both vandals and benign users have similar editing behavior*: Figure 3.2(c) shows just the first 4 edits made by both vandals and benign users. We see here that the percentage of re-edits and consecutive edits are almost the same in both cases.

3.3.2 Differences between Vandals and Benign User Behavior (w/o reversion features)

We also identify several behaviors which differentiate between vandals and benign users.

• *Vandals make faster edits than benign users.* On average, vandals make 35% of their edits within 15 minutes of the previous edit while benign users make 29.79% of their edits within 15 minutes (Figure 3.2(d)). This difference is statistically significant with a p-value of 8.2×10^{-82}.

•*Benign users spend more time editing a new (to them) page than vandals.* Vandals make 70% of their edits to a page they have not edited before within 15 minutes of their last edit, while for benign users the number is 54.3% (Figure 3.2(d)). This may be because a benign user must absorb the content of a new page before making thoughtful edits, while a vandal knows what he wants to say in advance and just goes ahead and says it.

• *The probability that benign users edit a meta-page is much higher than the same probability in the case of vandals.* Figure 3.2(e) shows that even in their very first edit, benign users have a 64.77% chance of editing a meta-page, while this is just 10.34% for vandals. If we look at the first 4 edits, the percentage of edits that are on meta-pages is 62% for benign users and just 11.1% for vandals. And if we look at all the edits, 40.72% of edits by normal users are on meta-pages, while only 21.57% of edits by vandals are on meta-pages.

3.3.3 Differences between Vandals and Benign User Behavior (including reversion)

For the sake of completeness, we also analyze the data looking for differences between vandal and benign user behavior when reverts are considered — however these differences are not considered in our vandal prediction methods.

• *Vandals make more edits driven by reversion than benign users.* Whenever a vandal u re-edits a page p, in 34.36% of the cases, u's previous edit on p was reverted by others. This almost never occurs in the case of benign users — the probability is just 4.8%. This suggests that

benign users are much more accepting of reversions than vandals.

•*The probability that a re-edit by a benign user of a page is accepted, even if previous edits by him on the same page were reverted, is much higher than for vandals*. Consider the case when a user edits a page *p* after some of his prior edits on *p* were reverted by other. If the user *u* is a benign user, it is more likely that his last edit is accepted. This suggests that the sequence of edits made by *u* were collaboratively edited by others with the last one surviving, suggesting that *u*'s reverts were constructive and were part of a genuine collaboration. Among the cases when *u* re-edits a page after one of his previous edits on *p* has been reverted, 89.87% of these re-edits survive for benign users, while this number is only 32.2% for vandals.

• *Vandals involve themselves in edit wars much more frequently than benign users*. A user u is said to participate in an edit war if there is a consecutive sequence of edits by u on the same page which is reverted at least two or three times (we consider both cases). Figure 3.2(f) shows that 27.9% of vandals make two pairs of consecutive edits because their previous edit was reverted, but only 13.41% of benign users do so. 12% of vandals make three such pairs of consecutive edits, compared to 2.9% in the case of benign users.

• *The probability that benign users discuss their edits is much higher than the probability of vandals doing so*. In 31.3% of the cases when a benign user consecutively edits a page p twice (i.e. the user is actively editing a page), he then edits a meta page. With vandals, this probability is 11.63%. This suggests that benign editors discuss edits on a meta-page after an edit, but vandals do not (perhaps because doing so would draw attention to the vandalism). In addition there is a 24.41% probability that benign users will re-edit a normal Wikipedia page after editing a meta-page while this happens much less frequently for vandals (only 6.17% vandals do such edits). This indicates that benign users, after discussing relevant issues on meta pages, edit a normal Wikipedia page.

• *Benign users consecutively surface edit pages a lot*. We define a surface edit by a user u on page p as: i) a consecutively edit on the same page p twice by u, and ii) the edit is not triggered by u's previous edit on p being reverted, and iii) made within 3 minutes of the previous edit by u. 50.94% benign users make at least one surface edit on a meta page, while only 8.54% vandals do so. On normal pages, both benign and normal users make such edits - there are 37.94% such cases for benign users and 36.94% for vandals. Over all pages, 24.24% benign users make at least 3 consecutive surface edits not driven by reversion, but only 7.82% vandals do so.

In conclusion: (i) Vandals make edits at a faster rate than benign users. (ii) Vandals are much less engaged in edits of meta pages, i.e. they are less involved in discussions with the community.

3.4 Vandal Prediction Algorithm

In the following sections, we use the insights from the previous section to classify vandals and benign users. Our vandal prediction methods use multiple known classifiers (SVM, decision trees, random forest and k-nearest neighbors) with different sets of features. In the accuracies reported in this section, the results are computed with SVM, as it gives the highest accuracy as reported in Section 6 using a 10-fold cross validation. *All features used for vandal prediction are behavior based and include no human generated revert information whatsoever. Thus, these approaches form an early warning system for Wikipedia administrators.*

3.4.1 Wikipedia Vandal Behavior (WVB) Approach

WVB uses features derived from consecutive edits. These are found by frequent pattern mining of the user_log dataset. Specifically, we extract the frequent patterns on both benign and vandal user logs – then, for each frequent pattern of benign users, we compute the frequency of the same pattern for vandals and vice versa. Finally, we select the patterns having significant

frequency difference between the two classes. The resulting features are described below:

1. **Consecutive re-edit, slowly (crs)**: whether or not the user edited the same page consecutively with a time gap exceeding 15 minutes.

2. **Consecutive re-edit, very fast (crv)**: whether or not the user edited the same page consecutively and less than 3 minutes passed between the two edits.

3. **Consecutive re-edit of a meta-page (crm)**: the number of times the user re-edited the same meta-page, consecutively.

4. **Consecutive re-edit of a non-meta-page (crn)**: whether or not the user re-edited the same non-meta-page, consecutively.

5. **Consecutive re-edit of a meta-page, very fast (crmv)**: whether or not the user re-edited the same meta-page, consecutively, and less than 3 minutes passed between the two edits.

6. **Consecutive re-edit of a meta-page, fast (crmf)**: whether or not the user re-edited the same meta-page, consecutively, and 3 to 15 minutes passed between the two edits.

7. **Consecutive re-edit of a meta-page, slowly (crms)**: whether or not the user re-edited the same meta-page, consecutively, and more than 15 minutes passed between the two edits.

8. **Consecutively re-edit fast and consecutively re-edit very fast (crf_crv)**: whether or not the following pattern is observed in the user log - the user re-edited the same article within 15 minutes, and later re-edited a (possibly different) article and less than 3 minutes passed between the second pair of edits.

9. **First edit meta-page (fm)**: whether or not the first edit of the user was on a meta-page. This in itself is quite a distinguishing feature, because usually vandals first edit a non-meta page and benign users first edit a meta-page. Therefore, this becomes quite an important feature for distinguishing the two.

10. **Edit of a new page at distance at most 3 hops, slowly (ntus)**: whether or not the user

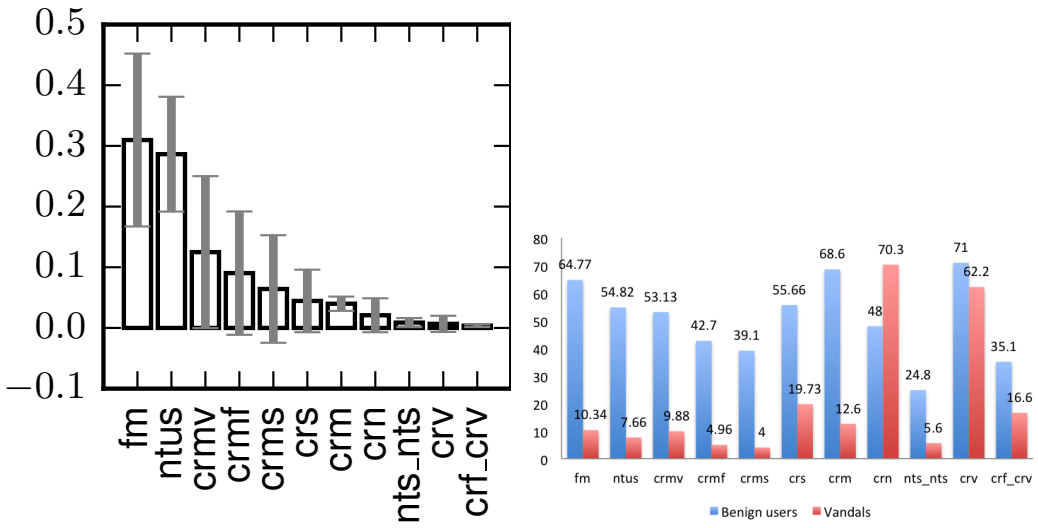
edited a new page (never edited by him before) p_2 which is within 3 hops or less of the previous page p_1 that he edited and either p_1 or p_2's category is unknown[6] and the time gap between the two edits exceeds 15 minutes.

11. **Edit of a new page at distance at most 3 hops slowly and twice (nts_nts)**: whether or not there are two occurrences of the following feature in the user log: *Edit of a new page at distance at most 3 hops, slowly (nts)*, i.e. in a pair (p_1, p_2) of consecutive edits, whether or not the user edited a new page p_2 (i.e. never edited before) such that p_2 can be reached from p_1 link-wise with at most 3 hops, and more than 15 minutes passed between the edit of p_1 and p_2.

In predicting vandals, we do not use any feature involving human identification of vandals (e.g. number of edits and reversion) because number of edits made has a bias towards benign users as they tend to perform more edits, while vandals perform fewer edits because they get blocked. Any feature that has a negative human intervention (number of reversions, number of warnings given to the user on a talk page, etc.) already indicates human recognition that a user may be a vandal. We explicitly avoid such features so that we provide Wikipedia administrators with a fully automated vandal early warning system.

Feature importance: We compute the importance of the features described above by using the fact that the depth of a feature, which is used as a node in a decision tree, captures the relative importance of that feature with respect to the target variable. Features at the top of the tree contribute to the final prediction decision of a larger fraction of inputs. The expected fraction of samples they contribute to can be used to estimate their importance. Figure 3.3(a) shows the importance of the different features for the classification task, which was computed by using a forest of 250 randomized decision trees (extra-trees [112]). The red bars in the plot show the

[6]This happens mostly for meta-pages though it can occasionally also happen for normal (non-meta) pages.



(a) (b)

Figure 3.3: (a) Importance of features (w/o reversion), and (b) Percentage of vandals and benign users with particular features (w/o reversion).

feature importance using the whole forest, with their variability across the trees represented by the blue bars. From the figure, it is clear that the features - *fm*, *ntus* and *crmv* - are the three most descriptive features for the classes. These are shown in greater detail in Figure 3.3(b). Let us look into each of them one by one.

• *If the very first page edited by user u is a normal (non-meta) page, then u is much more likely to be a vandal (64.77%) than a benign user (10.34%).* The fm feature tells us that when a user's first edit is on a normal page, the user is much more likely to be a vandal.

• *Benign users are likely to take longer to edit a new page than a vandal ($ntus$).* The probability that a benign user takes more than 15 minutes to edit the next page in an edit pair (p_1, p_2) when p_2 is within 3 hops of p_1, and p_1 or p_2's category is unknown is much higher (54.82%) than for vandals (7.66%). This suggests that benign users take longer to edit pages than vandals, possibly because they are careful and anxious to do a good job. Moreover, as p_1 or p_2 have no categories, the page is more likely to be a meta-page.

• *Benign users are much more likely to re-edit the same meta-page quickly (within 3 minutes) than vandals.* This usually happens when there is a minor mistake on the page, and the user edits to correct it. Note that this again has the feature that the edit was made on a meta page. Benign users are much more likely to make such edits (53.13%) than vandals (9.88%).

The top three features indicate that editing meta versus normal Wikipedia pages is a strong indicator of whether the user is benign. Intuitively, vandals vandalize heavily accessed pages and so normal pages are their most common target. On the other hand, benign users interact and discuss issues with other users about the content of the edit, and this discussion is done on meta pages.

Accuracy: Using an SVM classifier, the WVB approach obtains an accuracy of 86.6% in classifying Wikipedia users as vandals or benign on our entire user_log dataset.

3.4.2 Wikipedia Transition Probability Matrix (WTPM) Approach

The Wikipedia Transition Probability Matrix (WTPM) captures the edit summary of the users. The states in WTPM correspond to the space of possible vectors of features associated with any edit pair (p_1, p_2) carried out by a user u. By looking at Table 3.1, we see that there are 2 options for whether p_2 is a meta-page or not, 3 options for the time difference between edits (p_1, p_2), and so forth. This gives us a total of 60 possible states. Example states include: *consecutively re-edit a normal-page within 15 minutes* (s_1), or *edit a new normal page* p_2 *within 3 hops from* p_1 *and no common categories within 3 minutes* (s_2), etc.

The transition matrix $T(u)$ of user u captures the probability $T_{ij}(u)$ that user u goes from state s_i to s_j. $T_{ij} = \frac{N(s_i,s_j)}{\sum_k N(s_i,s_k)}$, where $N(s_i, s_j)$ is the number of times the user went from state s_i to s_j. This gives a (usually sparse) transition matrix of size 60 $\times$ 60 = 3600.

The intuition behind using WTPM as features for classification is that the transition proba-

bility from one state to the other for a vandal may differ from that of a benign user. Moreover, the states visited by vandals may be different from states visited by benign users (for example, it turns out that benign users are more likely to visit a state corresponding to "first edit on meta page", than vandals do).

We create a compact and distributed representation of $T(u)$ using an auto-encoder [113] — this representation provides the features for our SVM classifier. When doing cross-validation, we train the auto-encoder using the training set with input from both benign users and vandals. We then take the value given by the hidden layer for each input as the feature for training a classifier. For predicting output for the test set, we give each test instance as input to the auto-encoder and feed its representation from the hidden layer into the classifier. Note that the auto-encoder is trained only on the training set, and the representation for the test set is derived only from this learned model.

Accuracy: With a neural net auto-encoder of 400 hidden units and with SVM as the classifier, the WTPM approach gives an accuracy of 87.39% on the entire dataset.

3.4.3 VEWS Algorithm

The VEWS approach merges all the features used by both the WVB approach and the WTPM approach. The resulting accuracy with a SVM classifier slightly improves the accuracy of classification to 87.82%.

3.5 Experimental Evaluation: Vandal Prediction

We used the popularly used machine learning library called Scikit-learn [114] for our experiments and the deep learning library Theano [115] for training the auto-encoder.

Experiment 1: Overall Classification Accuracy. Table 3.2 shows the overall classification

	Accuracy	TPR	TNR	FPR	FNR
WVB	86.6%	0.85	0.89	0.11	0.15
WTPM	87.39%	0.88	0.90	0.10	0.12
VEWS	87.82%	0.87	0.92	0.08	0.13

Table 3.2: Table showing the accuracy and statistical values derived from the confusion matrix for the three approaches, on the entire dataset and averaged over 10 folds (without reversion features). The positive and negative class represent benign and vandal users, respectively.

accuracy of all three approaches by doing a 10-fold cross validation using an SVM classifier, together with the true positive, true negative, false positive, and false negative rates. We see that TP and TN rates are uniformly high, and FP and FN rates are low, making SVM an excellent classifier.

We use McNemar's paired test to check whether the three approaches produce different results. For all pairs among the three approaches, the null hypothesis that they produce the same results is rejected with p-value < 0.01, showing statistical significance. Overall, VEWS produces the best result even though it has slightly lower true positives and slightly more false negatives than WTPM.

We also classified using the VEWS approach with decision tree classifier, random forest classifier (with 10 trees) and k-nearest neighbors classifier (with k = 3) which gave classification accuracy of 82.82%, 86.62% and 85.4% respectively. We experimented with other classifiers as well, but they gave lower accuracy.

Experiment 2: Improvement with Temporal Recency. The previous experiment's cross validation randomly selects samples from the entire dataset for training and validation. But in the real world, a vandal's behavior may be more closely related to other vandals' recent behavior. To check this, starting from April 2013, for each month m, we train our algorithms with data from

all the users who started editing on Wikipedia within the previous three months, i.e. in months $m-3, m-2$ and $m-1$. m is varied until July 2014. We then use the learned model to predict whether a user is vandal or benign among the users who made their first edit in month m. The variation of accuracy is shown in Figure 3.4. The highest accuracy of 91.66% is obtained with the VEWS approach, when predicting for users who started editing in January 2014 and training is done with users from October-December 2013. The average accuracy for the three approaches over all the time points is also shown in Figure 3.4.

The most important observation from Figure 3.4 is that temporal classification accuracy for each approach is usually higher than the base accuracy shown in Table 3.2 and Figure 3.6 (described later in Experiment 4). We attribute this to the fact that in the previous experiment, we use cross-validation without considering temporal information when creating the folds. This experiment, on the other hand, predicts vandals based on what is learned during the previous three months.

Figure 3.4 shows that the approaches are consistent over time in separating vandals from benign users. At all times, the approaches have at least 85% classification accuracy, with the exception of the case when using WVB during months May and June, 2013.

Experiment 3: Varying Size of Training Set on Classification Accuracy. We designed an experiment to study the affect of varying the size of the training set, while maintaining the temporal aspect intact. So for testing on users who made their first edit in the month of July 2014, we train the classifier on edits made by users who started editing in the previous n months. We vary n from 1 to 12. This preserves the temporal aspect in training, similar to the previous experiment. The variation of accuracy is shown in Figure 3.5. There are two interesting observations: i) the accuracy of WTPM and VEWS increases with the number of (training) months n. ii) In contrast, WVB's accuracy is hardly affected by the number of months of training data. This is because: (i)

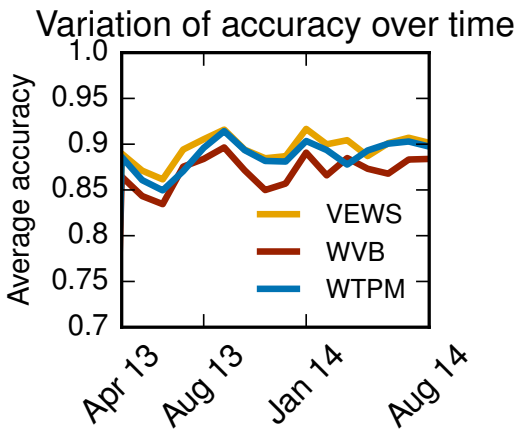


Figure 3.4: Plot showing variation of accuracy when training on edit log of users who started editing within previous 3 months (without reversion features). The table reports the average accuracy of all three approaches.

features in WVB are binary and (ii) *fm*, which is the most important feature in WVB, does not vary with time.

Experiments 2 and 3 show strong temporal dependency of user behavior on prediction of vandals. This may be due to several factors: Wikipedia may change rules and policies that affect user behavior, real world events might trigger users to make similar edits and emulate similar behaviour, etc. Such behavior traits would be highlighted when observing recent edits made by newly active users.

Experiment 4: Effect of First k User Edits. We study the effect of the first-k edits made by the user on prediction accuracy which is averaged over 10 folds of the whole dataset. The solid lines in Figure 3.6 show the variation in accuracy when k is varied from 1 to 500. As there is little shift in classification accuracy when $k > 20$, the situation for $k = 1, \ldots, 20$ is highlighted. We get an average accuracy of 86.6% for WVB, 87.39% for WTPM, and 87.82% for VEWS when k = 500. It is clear that the first edit itself (was the first edit made on a meta-page or not?) is a very strong classifier, with an accuracy of 77.4%. Accuracy increases fast when k is increased to

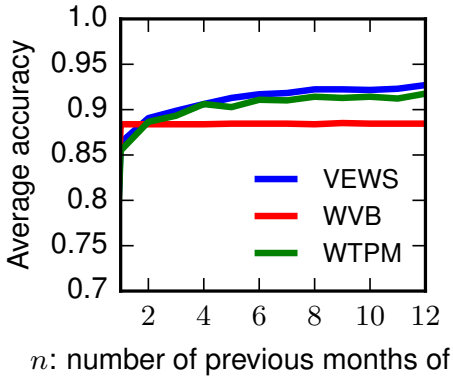


Figure 3.5: Plot showing the change in accuracy by varying the training set of users who started editing Wikipedia at most n months before July 2014. The testing is done on users who started editing in July 2014.

10 for all approaches, after which it flattens out. This suggests that a user's first few edits are very significant in deciding whether he/she is vandal or not.

Considering reversion. Figure 3.6 also shows that accuracy does go up by about 2% when we allow our three algorithms to consider reversion information. *Please note that this experiment is merely for completeness sake and our proposed algorithm does not depend on reversion at all.* For this experiment, we added additional reversion-driven edit features to the features used by WVB, WTPM, and VEWS (and we called these approaches WVB-WR, WTPM-WR, and VEWS-WR, respectively). These features capture whether a user edited a page after his previous edit on that page was reverted. Specifically, we extend the features - *crs, crv, crm, crn, crmv, crmf, crms and crf_crv* - to now have two types of re-edits: one that is reversion driven and one that is not. Using reversion information would mean that a human or vandalism detection system has already flagged a potential vandal. In contrast, our algorithms are able to predict vandals with high accuracy even without such input.

Comparison with State-of-the-art tools. Here we evaluate our work against ClueBot NG [103] and STiki [104] as they are the primary tools currently used by Wikipedia to detect

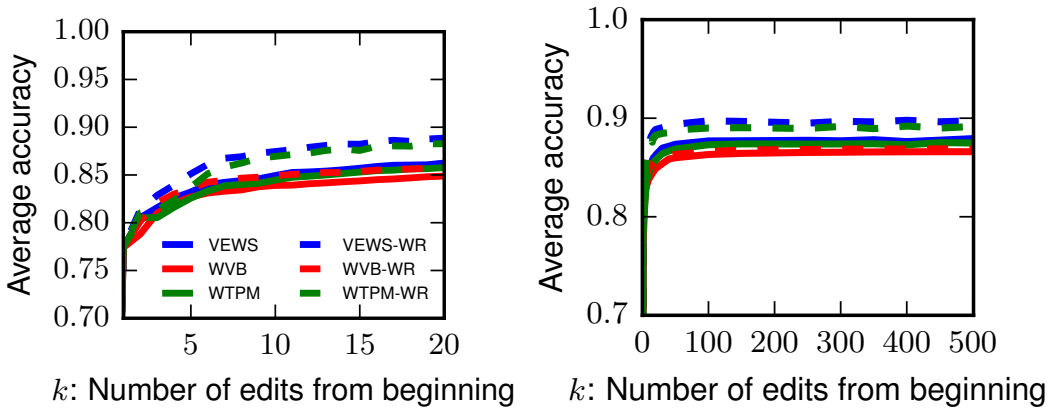


Figure 3.6: Plot showing variation of accuracy with the number of first k edits. The outer plot focuses on the variation of k from 1 to 20. The inset plot shows variation of k from 1 to 500.

vandalism. We recall that these tools are designed to detect whether the content of an article has been vandalized or not, while we focus on detecting whether a user is a vandal or not. We show that VEWS handily beats both ClueBot NG and Stiki in the latter task. Interestingly, when we combine VEWS', ClueBot NG's and STiki's features, we get better accuracy than with either of them alone. All experiments are done using 10-fold cross validation and SVM as the classifier.

Comparison with ClueBot NG. Given an edit, ClueBot NG [103] detects and reverts vandalism automatically. We could use ClueBot NG to classify a user as a vandal if he has made at least v vandalism edits (edits that were reverted by ClueBot NG). For comparing this heuristic with VEWS we use $v = 1, 2, 3$. Figure 3.7 shows that the maximum accuracy achieved by ClueBot NG is 71.4% (when v = 1) and accuracy decreases as v increases. Therefore, VEWS outperforms this use of ClueBot NG.

When does VEWS Detect Vandals? Of 17,027 vandals in our dataset, VEWS detects 3,746 that ClueBot NG does not detect (i.e. where ClueBot NG does not revert any edits by this person). In addition, it detects 7,795 vandals before ClueBot NG – on average 2.6 edits before ClueBot NG did. In 210 cases, ClueBot NG detects a vandal edit 5.29 edits earlier (on average) than VEWS

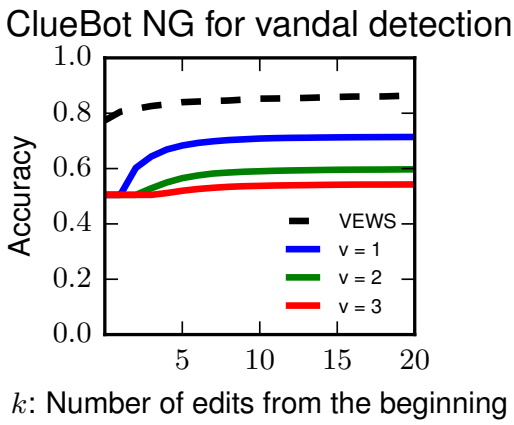


Figure 3.7: Plot showing the variation of accuracy for vandal detection by considering reversions made by ClueBot NG.

detects the vandal and there are 1,119 vandals that ClueBot NG detects but VEWS does not. Overall, when both detect the vandal, VEWS does it 2.39 edits (on average) before ClueBot NG does.

Instead of reverts made by ClueBot NG, when we consider reverts made by any human or any known vandalism detection system, VEWS detects the vandal at least as early as its first reversion in 87.36% cases — in 43.68% of cases, VEWS detects the vandal 2.03 edits before the first reversion. *Thus, on aggregate, VEWS outperforms both humans and other vandalism detection system in early detection of vandals*, though there are definitely a small number of cases (7.8%) on which ClueBot NG performs very well.[7]

Comparison with STiki. STiki provides a "probability of vandalism" score to each edit. STiki also maintains a user reputation score, which is developed by looking at the user's past edits (the higher the score, the higher the probability that the user is a vandal). We use both these scores separately to compare against STiki.

We first consider a user to be a vandal if his STiki reputation score (REP_USER) after making the k^{th} edit is at least t. Figure 3.8 shows the results of this experiment where we vary

[7]We do not compare with STiki in this experiment, as it does not automatically revert edits.

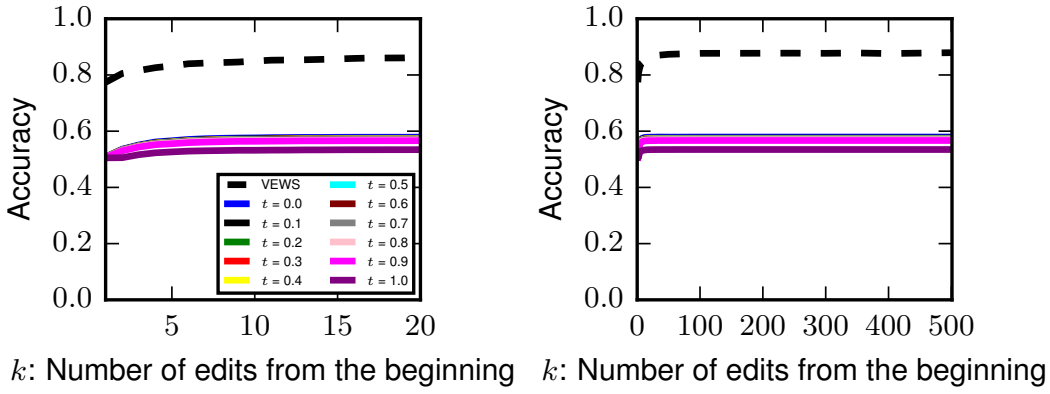


Figure 3.8: Plot showing the variation of accuracy for vandal detection by considering k^{th} REP_USER score given by STiki.

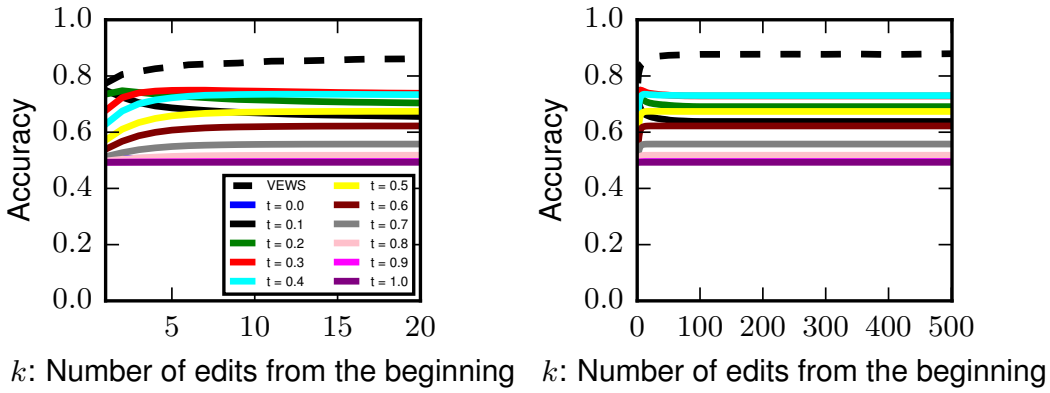


Figure 3.9: Plot showing the variation of accuracy for vandal detection by considering article scores given by STiki. RULE: If the user makes 1 edit in first k that gets score $> t$, then the user is a vandal.

t from 0 to 1 in steps of 0.1. We also report the VEWS curve for comparison. We see that the STiki user reputation score to detect vandals has less than 60% accuracy and is handily beaten by VEWS. We do not test for values of t greater than 1 as accuracy decreases as t increases.

In the second experiment, we say that a user is a vandal after making k edits if the maximum STiki score among these k edits is more than a threshold t.[8] We vary the values of t from 0 to 1 and the results can be seen in Figure 3.9. We also did experiments for the case when we classify a

[8]We also tested using average instead of maximum with similar results.

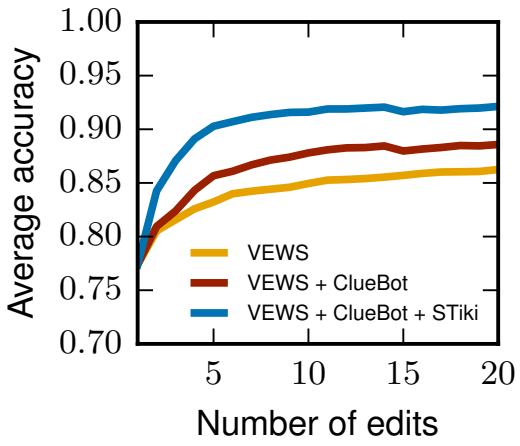


Figure 3.10: Figure showing effect of adding STiki and ClueBot NG's features to our VEWS features.

user as a vandal if the maximum two and three scores are above t, which yielded lower accuracy scores.

Combining VEWS, Cluebot NG and STiki. VEWS can be improved by adding linguistic and meta-data features from ClueBot NG and STiki. In addition to the features in VEWS, we add the following features: i) number of edits reverted[9] by ClueBot NG until the k^{th} edit, ii) user reputation score by STiki after the k^{th} edit, and iii) maximum article edit score given by STiki until the k^{th} edit (we also did experiments with the average article edit score instead of maximum, which gave similar results). Figure 3.10 shows the variation of average accuracy by using the first-k edits made by the user to identify it as a vandal. The accuracy of the VEWS-ClueBot combination is 88.6% ($k = 20$), which is higher than either of them alone. Observe that this combination does not consider any human input. The accuracy of the combination VEWS-ClueBot-STiki improves slightly to 90.8% ($k = 20$), but STiki considers human inputs while calculating its scores.

[9]We allow these reverts to be considered as they are generated with no human input, so the resulting combination is still fully automated.

3.6 Conclusions

In this chapter, we looked at the problem of vandals in Wikipedia. We developed a theory based on edit-pairs and edit-patterns to study the behavior of vandals on Wikipedia and distinguish these behaviors from those of benign users. We made the following contributions.

1. First, we developed the UMDWikipedia dataset which contains a host of information about Wikipedia users and their behaviors.

2. Second, we conducted a detailed analysis of behaviors that distinguish vandals from benign users. Notable distinctions that do not involve revert information include:

(a) We found that the first page edited by vandals is much more likely to be a normal page – in contrast, benign users' first edits are much more likely to occur on meta-pages.

(b) We found that benign users take longer to edit a page than a vandal user.

(c) We found that benign users are much more likely to re-edit the same page quickly (within 3 minutes) as compared to vandals, possibly because they wanted to go back and improve or fix something they previously wrote.

These three major findings, along with a few more, helped us to provide some of the first insights into the behavior of vandals and benign editors, that do not depend on reverts that differentiate between vandals and benign users.

3. We developed three approaches to predict which users are vandals. Each of these approaches uses SVM with different sets of features. Our VEWS algorithm provides the best performance, achieving 87.82% accuracy. If in addition we consider temporal factors, namely that vandals next month are more likely to behave like vandals in the last few months, this accuracy goes up to 89.5%. Moreover, we show that the combination of VEWS and past work (ClueBot NG and STiki) increases accuracy to 90.8%, even without any human generated reversion infor-

mation. Moreover, VEWS detects far more vandals than ClueBot NG. When both VEWS and ClueBot NG predict vandals, VEWS does it 2.39 edits (on average) before ClueBot NG does.

Overall, we found several similarities and differences in the behavior of vandal and benign editors on Wikipedia. Based on these differences, we develop an early warning system that predicts vandals efficiently from the first few edits of the editor.

Chapter 4

Detecting Hoax Articles in Wikipedia

In this chapter, we study the presence of false information in Wikipedia in terms of hoax articles. As we studied in the previous chapter, the presence of vandals harms the integrity of Wikipedia. While vandals make edits that may be blatantly malicious, hoax articles are carefully crafted to masquerade as the truth. Some hoax articles are so well created that they survive for years without being detected. To spot such false information, we first study the properties of hoax articles, understand how they differ from legitimate articles, and then create a machine learning model to identify the hoax articles.

4.1 Introduction

False information is abundant on the web, ranging from satirical news websites that publish often-harmless fake news to scams that often lead to serious financial consequences. The reasons for communicating false information vary widely: on the one extreme, *misinformation* is conveyed in the honest but mistaken belief that the relayed incorrect facts are true; on the other extreme, *disinformation* denotes false facts that are conceived in order to deliberately deceive or betray an audience [116, 117]. A third class of false information has been called *bullshit,* where the agent's

primary purpose is not to mislead an audience into believing false facts, but rather to "convey a certain impression of himself" [118].

All these types of false information are abundant on the Web, and regardless of whether a fact is fabricated or misrepresented on purpose or not, the effects it has on people's lives may be detrimental and even fatal, as in the case of medical lies [68, 119–121].

This chapter focuses on a specific kind of disinformation, namely ***hoaxes***. Wikipedia defines a hoax as "a deliberately fabricated falsehood made to masquerade as truth." The Oxford English Dictionary adds another aspect by defining a hoax as "a *humorous* or mischievous deception" (italics ours).

We study hoaxes in the context of Wikipedia, for which there are two good reasons: first, anyone can insert information into Wikipedia by creating and editing articles; and second, as the world's largest encyclopedia and one of the most visited sites on the Web, Wikipedia is a major source of information for many people. In other words: Wikipedia has the potential to both attract and spread false information in general, and hoaxes in particular.

The impact of some Wikipedia hoaxes has been considerable, and anecdotes are aplenty. The hoax article about a fake language called "Balboa Creole French", supposed to be spoken on Balboa Island in California, is reported to have resulted in "people coming to [...] Balboa Island to study this imaginary language" [122]. Some hoaxes have made it into books, as in the case of the alleged (but fake) Aboriginal Australian god "Jar'Edo Wens", who inspired a character's name in a science fiction book [123] and has been listed as a real god in at least one nonfiction book [124], all before it came to light in March 2015 that the article was a hoax. Another hoax ("Bicholim conflict") was so elaborate that it was officially awarded "good article" status and maintained it for half a decade, before finally being debunked in 2012 [125].

The list of extreme cases could be continued, and the popular press has covered such in-

cidents widely. What is less available, however, is a more general understanding of Wikipedia hoaxes that goes beyond such cherry-picked examples.

Our contributions: impact, characteristics, and detection of Wikipedia hoaxes. This chapter takes a broad perspective by starting from the set of all hoax articles ever created on Wikipedia and illuminating them from several angles. We study over 20,000 hoax articles, identified by the fact that they were explicitly flagged as potential hoaxes by a Wikipedia editor at some point and deleted after a discussion among editors who concluded that the article was indeed a hoax. Some articles are acquitted as a consequence of that discussion, and we study those as well.

When answering a question on the Q&A site Quora regarding the aforementioned hoax that had been labeled as a "good article", Wikipedia founder Jimmy Wales wrote that "[t]he worst hoaxes are those which (a) last for a long time, (b) receive significant traffic and (c) are relied upon by credible news media" [126]. Inspired by this assessment, our first set of questions aims to understand how impactful (and hence detrimental, by Wales's reasoning) typical Wikipedia hoaxes are by quantifying (a) how long they last, (b) how much traffic they receive, and (c) how heavily they are cited on the Web. We find that most hoaxes have negligible impact along all of these three dimensions, but that a small fraction receives significant attention: 1% of hoaxes are viewed over 100 times per day on average before being uncovered.

In the second main part of the chapter, our goal is to delineate typical characteristics of hoaxes by comparing them to legitimate articles. We also study how successful (*i.e.*, long-lived and frequently viewed) hoaxes compare to failed ones, and why some truthful articles are mistakenly labeled as hoaxes by Wikipedia editors. In a nutshell, we find that on average successful hoaxes are nearly twice as long as legitimate articles, but that they look less like typical Wikipedia articles in terms of the templates, infoboxes, and inter-article links they contain. Further, we find that the "wiki-likeness" of legitimate articles wrongly flagged as hoaxes is even lower than that of actual

hoaxes, which suggests that administrators put a lot of weight on these superficial features when assessing the veracity of an article.

The importance of the above features is intuitive, since they are so salient, but in our analysis we find that less immediately available features are even more telling. For instance, new articles about real concepts are often created because there was a need for them, reflected in the fact that the concept is mentioned in many other articles before the new article is created. Hoaxes, on the contrary, are mentioned much less frequently before creation—they are about nonexistent concepts, after all—but interestingly, many hoaxes still receive some mentions before being created. We observe that such mentions tend to be inserted shortly before the hoax is created, and by anonymous users who may well be the hoaxsters themselves acting *incognito*.

The creator's history of contributions made to Wikipedia before a new article is created is a further major distinguishing factor between different types of articles: most legitimate articles are added by established users with many prior edits, whereas hoaxes tend to be created by users who register specifically for that purpose.

Our third contribution consists of the application of these findings by building machine-learned classifiers for a variety of tasks revolving around hoaxes, such as deciding whether a given article is a hoax or not. We obtain good performance; *e.g.*, on a balanced dataset, where guessing would yield an accuracy of 50%, we achieve 91%. To put our research into practice, we finally find hoaxes that have not been discovered before by running our classifier on Wikipedia's entire revision history.

Finally, we aim to assess how good humans are at telling apart hoaxes from legitimate articles in a typical reading situation, where users do not explicitly fact-check the article by using a search engine, following up on references, etc. To this end, we design and run an experiment involving human raters who are shown pairs consisting of one hoax and one non-hoax and asked to

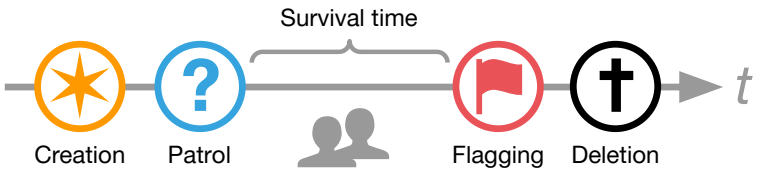


Figure 4.1: Life cycle of a Wikipedia hoax article. After the article is created, it passes through a human verification process called patrol. The article survives until it is flagged as a hoax and eventually removed from Wikipedia.

decide which one is the hoax by just inspecting the articles without searching the Web or following links. Human accuracy on this task is only 66% and is handily surpassed by our classifier, which achieves 86% on the same test set. The reason is that humans are biased to believe that well-formatted articles are legitimate and real, whereas it is easy for our classifier to see through the markup glitter by also considering features computed from other articles (such as the number of mentions the article in question receives) as well as the creator's edit history.

4.2 Data: Wikipedia hoaxes

The Wikipedia community guidelines define a hoax as "an attempt to trick an audience into believing that something false is real", and therefore consider it "simply a more obscure, less obvious form of vandalism" [29].

A distinction must be made between *hoax articles* and *hoax facts*. The former are entire articles about nonexistent people, entities, events, etc., such as the fake Balboa Creole French language mentioned in the introduction.[1] The latter are false facts about existing entities, such as

[1] Occasionally users create articles about existing unimportant entities and present them as important, as in the case of a Scottish worker who created an article about himself claiming he was a highly decorated army officer [127]. We treat these cases the same way as fully fabricated ones: whether Captain Sir Alan Mcilwraith never existed or exists but

the unfounded and false claim that American journalist John Seigenthaler "was thought to have been directly involved in the Kennedy assassinations" [128].

Finding hoax facts is technically difficult, as Wikipedia provides no means of tagging precisely one fact embedded into a mostly correct article as false. However, in order to find hoax articles, it suffices to look for articles that were flagged as such at some point. Hence we focus on hoax articles in this chapter.

To describe the mechanism by which hoax articles are flagged, we need to consider Wikipedia's page creation process (schematized in Fig. 4.1). Since January 2006 the privilege of creating new articles has been limited to logged-in users (*i.e.*, we know for each new article who created it). Once the article has been created, it appears on a special page that is monitored by trusted, verified Wikipedians who attempt to determine the truthfulness of the new article and either mark it as legitimate or flag it as suspicious by pasting a template[2] into the wiki markup text of the article.

This so-called patrolling process (introduced in November 2007) works very promptly: we find that 80% of all new articles are patrolled within an hour of creation, and 95% within a day. This way many suspicious articles are caught and flagged immediately at the source. Note that flagging is not restricted to patrol but may happen at any point during the lifetime of the article. Once flagged, the article is discussed among Wikipedians and, depending on the verdict, deleted or reinstituted (by removing the hoax template). The discussion period is generally brief: 88% of articles that are eventually deleted are deleted within a day of flagging, 95% within a week, and 99% within a month. We define the *survival time* of a hoax as the time between patrolling and flagging (Fig. 4.1).

In this chapter we consider as hoaxes all articles of the English Wikipedia that have gone through this life cycle of creation, patrol, flagging, and deletion. There are 21,218 such articles.

is in fact a Glasgow call center employee does not make a real difference for all intents and purposes.

[2] {{db-hoax}} for blatant, and {{hoax}} for less obvious, hoaxes.

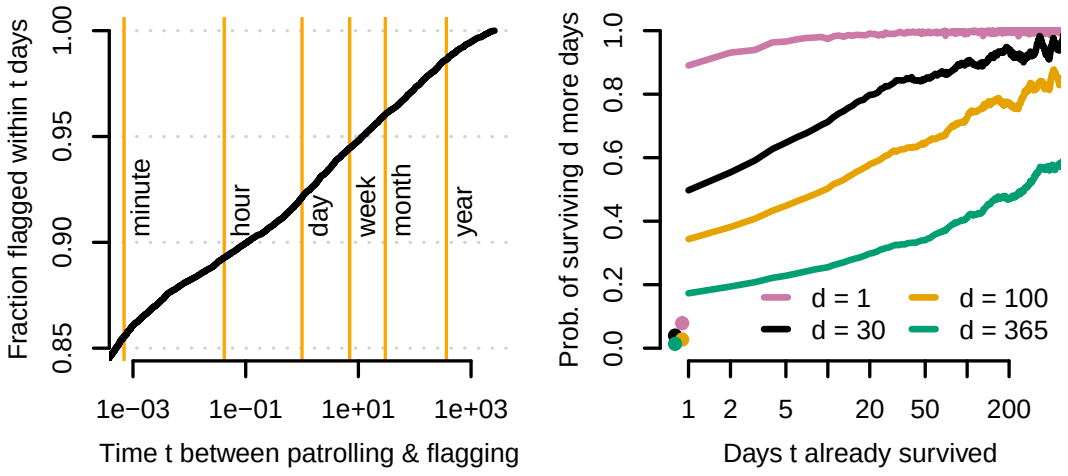
4.3 Real-world impact of hoaxes

Disinformation is detrimental if it affects many people. The more exposure hoaxes get, the more we should care about finding and removing them. Hence, inspired by the aforementioned Jimmy Wales quote that "[t]he worst hoaxes are those which (a) last for a long time, (b) receive significant traffic, and (c) are relied upon by credible news media" [126], we quantify the impact of hoaxes with respect to how long they survive (Sec. 4.3.1), how often they are viewed (Sec. 4.3.2), and how heavily they are cited on the Web (Sec. 4.3.3).

4.3.1 Time till discovery

As mentioned in Sec. 4.2, since November 2007 all newly created articles have been patrolled by trusted editors. Indeed, as shown by Fig. 4.2(a), most of the hoaxes that are ever discovered are flagged immediately at the source: *e.g.*, 90% are flagged within one hour of (so basically, during) patrol. Thereafter, however, the detection rate slows down considerably (note the logarithmic x-axis of Fig. 4.2(a)): it takes a day to catch 92% of eventually detected hoaxes, a week to catch 94%, a month to catch 96%, and one in a hundred survives for more than a year.

Next we ask how the chance of survival changes with time. For this purpose, Fig. 4.2(b) plots the probability of surviving for at least $t + d$ days, given that the hoax has already survived for t days, for $d = 1$, 30, 100, 365. Although the chance of surviving the first day is very low at only 8% (Fig. 4.2(a)), once a hoax has survived that day, it has a 90% chance of surviving for at least another day, a 50% chance of surviving for at least one more month, and an 18% chance of surviving for at least one more year (up from a prior probability of only 1% of surviving for at least a year). After this, the survival probabilities keep increasing; the longer the hoax has already survived, the more likely it becomes to stay alive.



(a) (b)

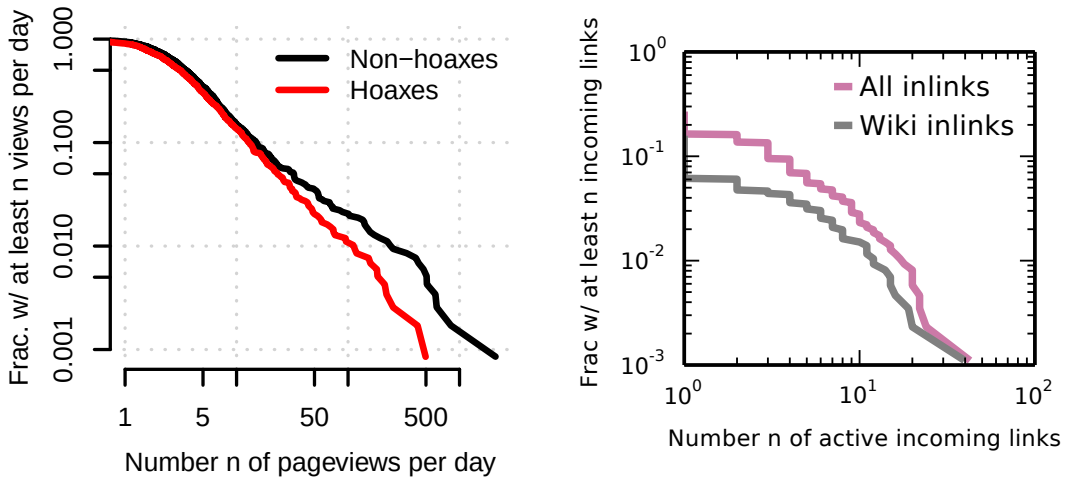
Figure 4.2: (a) Cumulative distribution function (CDF) of hoax survival time. Most hoaxes are caught very quickly. (b) Time the hoax has already survived on x-axis; probability of surviving d more days on y-axis (one curve per value of d). Dots in bottom left corner are prior probabilities of surviving for d days.

In summary, most hoaxes are very short-lived, but those that survive patrol have good odds of staying in Wikipedia for much longer. There is a relatively small number of longevous hoaxes, but as we show later, these hoaxes attract significant attention and a large number of pageviews.

4.3.2 Pageviews

Next we aim to assess the impact of Wikipedia hoaxes by studying pageview statistics as recorded in a dataset published by the Wikimedia Foundation and containing, for every hour since December 2007, how often each Wikipedia page was loaded during that hour [129].

We aggregate pageview counts for all hoaxes by day and normalize by the number of days the hoax survived, thus obtaining the average number of pageviews received per day between patrolling and flagging. Since this quantity may be noisy for very short survival times, we consider



(a) (b)

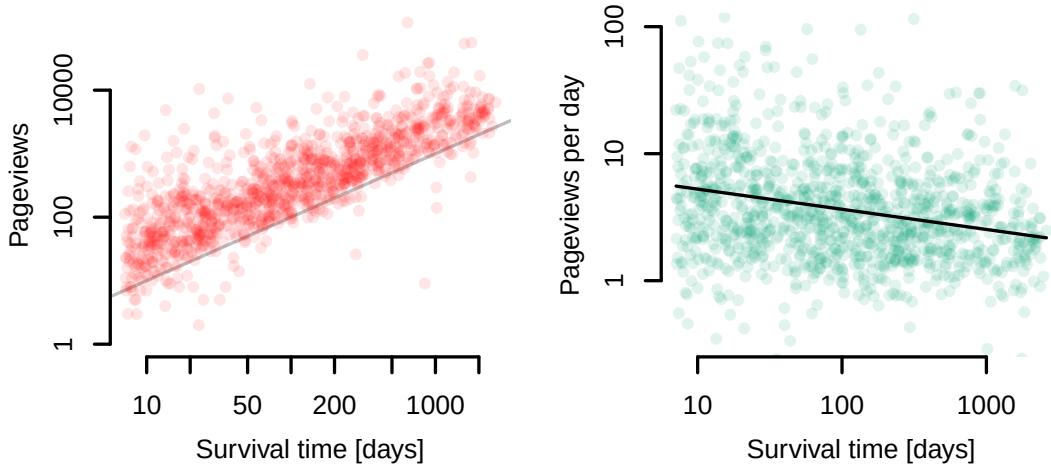
Figure 4.3: CCDFs of (a) number of pageviews for hoaxes and non-hoaxes (14% of hoaxes get over 10 pageviews per day during their lifetime) and (b) number of active inlinks from Web.

only hoaxes that survived for at least 7 days.[3] This leaves us with 1,175 of the original 21,218 hoaxes.

The complementary cumulative distribution function (CCDF) of the average number of pageviews per day is displayed as a red line in Fig. 4.3(a). As expected, we are dealing with a heavy-tailed distribution: most hoaxes are rarely viewed (median 3 views per day; 86% get fewer than 10 views per day), but a non-negligible number get a lot of views; *e.g.*, 1% of hoaxes surviving for at least a week get 100 or more views per day on average. Overall, hoaxes are viewed less than non-hoaxes, as shown by the black line in Fig. 4.3(a) (median 3.5 views per day; 85% get fewer than 10 views per day; for each hoax, we sampled one random non-hoax created on the same day as the hoax).

The facts that (1) some hoaxes survive much longer than others (Fig. 4.2(a)) and (2) some are viewed much more frequently per day than others (Fig. 4.3(a)) warrant the hypothesis that

[3] To avoid counting pageviews stemming from patrolling and flagging, we start counting days 24 hours after the end of the day of patrolling, and stop counting 24 hours before the start of the day of flagging.



(a) (b)

Figure 4.4: Longevous hoaxes are (a) viewed more over their lifetime (gray line $y = x$ plotted for orientation; not a fit) and (b) viewed less frequently per day on average (black line: linear-regression fit).

hoaxes might have a constant expected total number of pageviews until they are caught. This hypothesis would predict that plotting the total lifetime number of pageviews received by hoaxes against their survival times would result in a flat line. Fig. 4.4(a) shows that this is not the case, but that, instead, the hoaxes that survive longer also receive more pageviews.[4]

Finally, when plotting survival times against per-day (rather than total) pageview counts (Fig. 4.4(b)), we observe a negative trend (Spearman correlation -0.23). That is, pages that survive for very long receive fewer pageviews per day (and *vice versa*).

Together we conclude that, while there is a slight trend that hoaxes with more daily traffic

[4] It could be objected that this might be due to a constant amount of bot traffic per day (which is not excluded from the pageview dataset we use). To rule this out, we assumed a constant number b of bot hits per day, subtracted it (multiplied with the survival time) from each hoax's total count, and repeated Fig. 4.4(a) (for various values of b). We still observed the same trend (not plotted for space reasons), so we conclude that Fig. 4.4(a) is not an artifact of bot traffic.

generally get caught faster (Fig. 4.4(b)), it is not true that hoaxes are caught after a constant expected number of pageviews (Fig. 4.4(a)). It is not the case that only obscure, practically never visited hoaxes survive the longest; instead, we find that some carefully crafted hoaxes stay in Wikipedia for months or even years and get over 10,000 pageviews (24 hoaxes had over 10,000 views, and 375 had over 1,000 views).

4.3.3 References from the Web

Next we aim to investigate how different pages on the Web link and drive traffic to the hoax articles. While in principle there may be many pages on the Web linking to a particular Wikipedia hoax, we focus our attention on those links that are actually traversed and bring people to the hoax. To this end we utilize 5 months' worth of Wikipedia web server logs and rely on the HTTP referral information to identify sources of links that point to Wikipedia hoaxes.

In our analysis we only consider the traffic received by the hoax during the time it was live on Wikipedia, and not pre-creation or post-deletion traffic. There are 862 hoax articles that could potentially have received traffic during the time spanned by the server logs we use. We filter the logs to remove traffic that may have been due to article creation, patrol, flagging, and deletion, by removing all those requests made to the article during a one-day period around these events. This gives us 213 articles, viewed 23,353 times in total. Furthermore, we also categorize the different sources of requests into five broad categories based on the referrer URL: search engines, Wikipedia, social networks (Facebook and Twitter), Reddit, and a generic category containing all others. We define all search engine requests for an article as representing a single inlink. For the other categories, the inlink is defined by the URL's domain and path portions. We show the CCDF of the number of inlinks for the hoax articles in Fig. 4.3(b). On average, each hoax article has 1.1 inlinks. Not surprisingly, this distribution is heavily skewed, with most articles having no inlinks

Metric	SE	Wiki	SN	Reddit	Others
Average inlinks	0.78	2.1	0.08	0.15	1.3
Median inlinks	1	1	0	0	1
Inlinks per article	35%	29%	0.6%	3%	33%

Table 4.1: Number of inlinks per hoax article ("SE" stands for search engines, "SN" for social networks).

(median 0; 84% having at most one inlink). However, there is a significant fraction of articles with more inlinks; *e.g.*, 7% have 5 or more inlinks.

Table 4.1 gives the distribution of inlinks from different sources. Among the articles that have at least one inlink, search engines, Wikipedia, and "others" are the major sources of inbound connections, providing 35%, 29%, and 33% of article inlinks on average. These hoax articles have 2.1 inlinks from Wikipedia and 1.3 from "other" sources on average.

Overall, the analysis indicates that the hoax articles are accessible from multiple different locations, increasing the chances that they are viewed. Moreover, hoaxes are also frequently reached through search engines, indicating easy accessibility.

4.4 Characteristics of successful hoaxes

In the present section we attempt to elicit typical characteristics of Wikipedia hoaxes. In particular, we aim to gain a better understanding of (1) how hoaxes differ from legitimate articles, (2) how successful hoaxes differ from failed hoaxes, and (3) what features make a legitimate article be mistaken for a hoax.

To this end we compare four groups of Wikipedia articles in a descriptive analysis:

1. *Successful hoaxes* passed patrol, survived for significant time (at least one month from cre-

ation to flagging), and were frequently viewed (at least 5 times per day on average).

2. *Failed hoaxes* were flagged and deleted during patrol.
3. *Wrongly flagged articles* were temporarily flagged as hoaxes, but were acquitted during the discussion period and were hence not deleted.
4. *Legitimate articles* were never flagged as hoaxes.

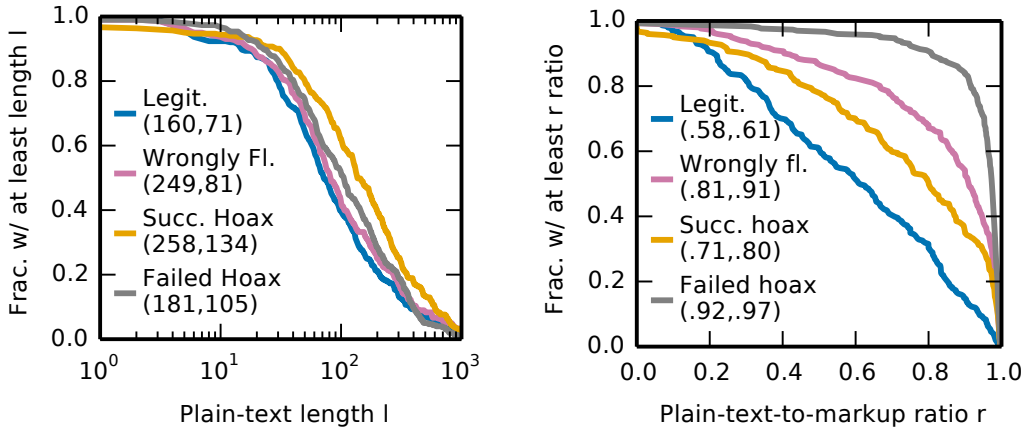
The set of all successful hoaxes consists of 301 pages created over a period of over 7 years. The usage patterns and community norms of Wikipedia may have changed during that period, and we want to make sure to not be affected by such temporal variation. Hence we ensure that the distribution of creation times is identical across all four article groups by subsampling an equal number of articles from each of groups 2, 3, and 4 while ensuring that for each successful hoax from group 1 there is another article in each group that was created on the same day as the hoax.

Given this dataset, we investigate commonalities and differences between the four article groups with respect to four types of features: (1) Appearance features (Sec. 4.4.1) are properties of the article that are immediately visible to a reader of the article. (2) Network features (Sec. 4.4.2) are derived from the so-called ego network formed by the other articles linked from the article in question. (3) Support features (Sec. 4.4.3) pertain to mentions of the considered article's title in other articles. (4) Editor features (Sec. 4.4.4) are obtained from the editor's activity before creating the article in question.

4.4.1 Appearance features

We use the term *appearance features* to refer to characteristics of an article that are directly visible to a reader.

Plain-text length. One of the most obvious features of an article is its length, which we define as the number of content words, obtained by first removing wiki markup (templates, images,



(a) Plain-text length

(b) Plain-text-to-markup ratio

Figure 4.5: CCDFs of appearance features; means and medians in brackets.

references, links to other articles and to external URLs, etc.) from the article source text and then tokenizing the resulting plain text at word boundaries.

Fig. 4.5(a) demonstrates that successful hoaxes are particularly long: their median number of content words is 134 and thus nearly twice as large as the median of legitimate articles (71). Further, and maybe surprisingly, failed hoaxes are the second most verbose group: with a median of 105 words, they are nearly 50% longer than legitimate articles.

Plain-text-to-markup ratio. While the length of an article is largely determined by the plain-text words it contains, the overall visual appearance of an article depends heavily on the wiki markup contained in the article source. The ratio of words after *vs.* before markup stripping may be taken as a measure of how "wiki-like" the article is: a ratio of 1 implies no wiki markup, *i.e.*, no wiki links to other articles, no infoboxes, references, images, footnotes, etc., and the article would look rather different from a typical Wikipedia article; the smaller the ratio, the more effort was devoted to making the article conform to Wikipedia's editorial standards.

Fig. 4.5(b) reveals striking differences between article groups. On one extreme, legitimate

articles contain on average 58% plain text. On the other extreme, failed hoaxes consist nearly entirely of plain text (92% in the mean). Successful hoaxes and wrongly flagged articles take a middle ground.

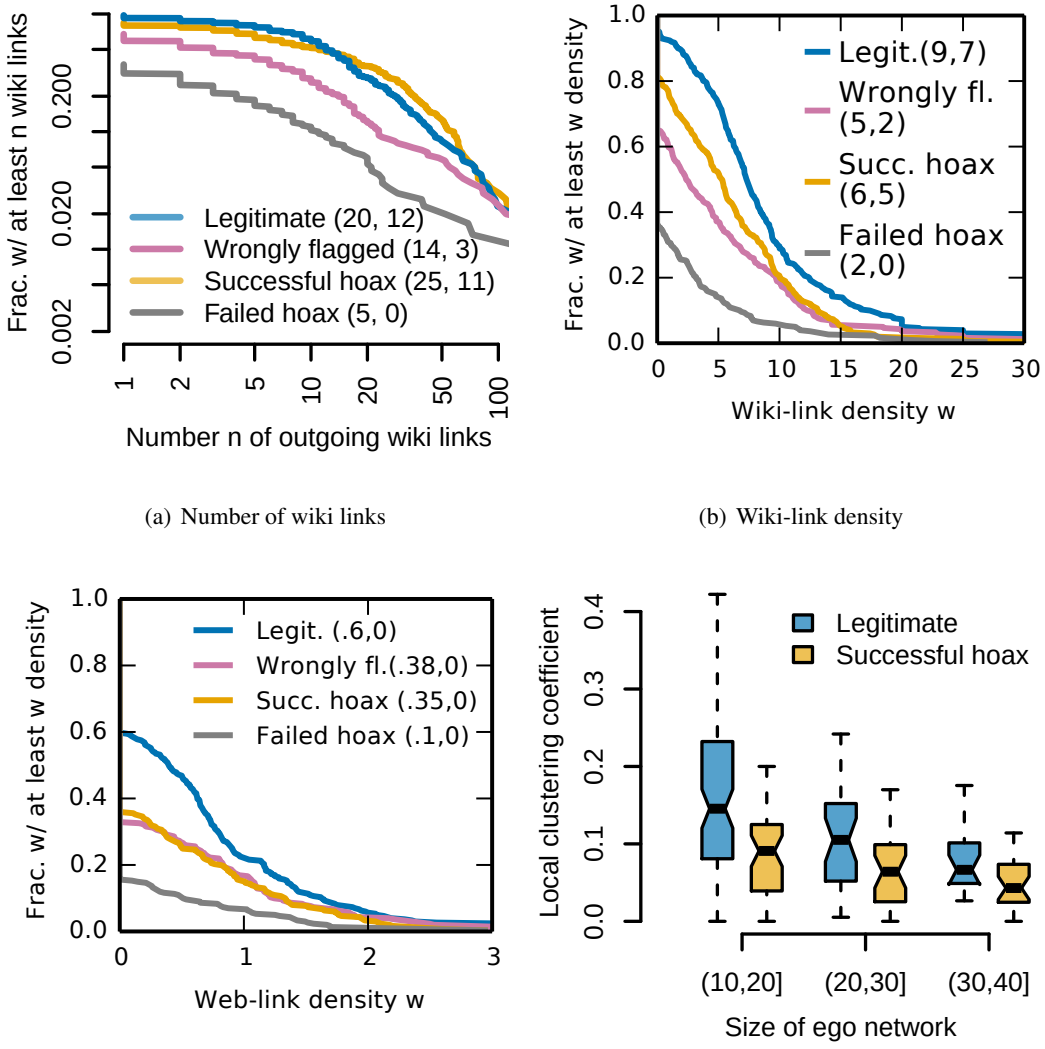
This suggests that embellishing a hoax with markup increases its chances of passing for legitimate and that, conversely, even legitimate articles that do not adhere to the typical Wikipedia style are likely to be mistaken for hoaxes. It is not so much the amount of bare-bones content that matters—wrongly flagged articles (median 81 words; Fig. 4.5(a)) are similarly long to unflagged legitimate articles (median 71)—but rather the amount of mixed-in markup.

Wiki-link density. Links between articles (so-called *wiki links*) constitute a particularly important type of markup. Legitimate articles and successful hoaxes contain similar numbers of wiki links (Fig. 4.6(a); medians 12 and 11, respectively); failed hoaxes and wrongly flagged articles, on the other hand, are much less well connected: their medians are only 0 and 3, respectively.

While the number of wiki links is similar for legitimate articles and hoaxes, we saw previously that successful hoaxes are nearly twice as long as legitimate articles on average. Hence another interesting measure is the density of wiki links, defined here as the number of wiki links per 100 words (counted before markup stripping because wiki links may be embedded into markup such as templates).

Under this measure the picture changes: as evident in Fig. 4.6(b), successful hoaxes have significantly fewer outlinks per 100 words than legitimate articles (medians 5 *vs.* 7). Wrongly flagged articles (median 2) look again more like hoaxes than legitimate articles, which is probably a contributing factor to their being suspected to be hoaxes.

Web-link density. Considering the density of links to general Web resources (Fig. 4.6(c)), rather than to other Wikipedia articles, results in the same conclusion that legitimate articles are considerably better referenced than articles that are, or are mistaken for, hoaxes.



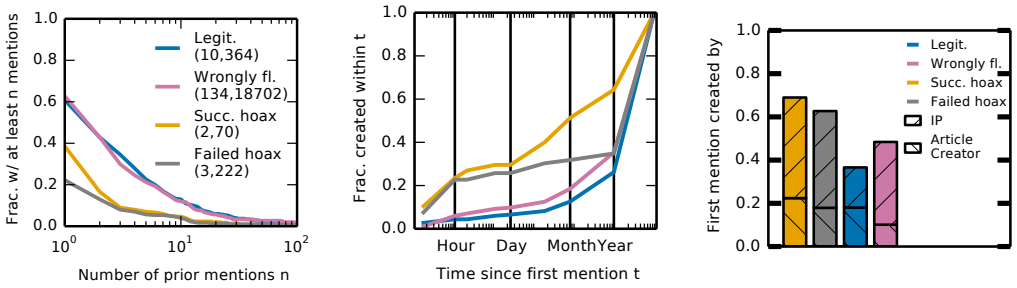
(a) Number of wiki links

(b) Wiki-link density

(c) Web-link density

(d) Ego-net clustering coefficient

Figure 4.6: Link characteristics: CCDFs (means/medians in brackets) of (a) number of wiki links, (b) wiki-link density, and (c) Web-link density. (d) Ego-network clustering coefficient as function of ego-network size (nodes of outdegree at most 10 neglected because clustering coefficient is too noisy for very small ego networks; nodes of outdegree above 40 neglected because they are very rare).



(a) Prior mentions (b) 1st prior mention (c) 1st-men. creator

Figure 4.7: Support features: (a) CCDF of number of mentions prior to article creation (means/ medians in brackets). (b) CDF of time from first prior mention to article creation. (c) Probability of first prior mention being inserted by hoax creator or anonymous user (identified by IP address), respectively.

4.4.2 Link network features

Above we treated features derived from embedded links as appearance features, since the links are clearly visible to a reader of the article. But they are at the same time features of the hyperlink network underlying Wikipedia. While outlinks constitute a first-order network feature (in the sense that they deal only with direct connections to other articles), it is also interesting to consider higher-order network features, by looking not only at what the article is connected to, but also how those connected articles are linked amongst each other.

Ego-network clustering coefficient. To formalize this notion, we consider an article A's *ego network,* defined as the undirected graph spanned by the articles linked from A and the links between them (A itself is not included in the ego network). Given the ego network, we compute its clustering coefficient [130] as the actual number of edges in the ego network, divided by the number of edges it would contain if it were fully connected.

Fig. 4.6(d) shows that legitimate articles tend to have larger clustering coefficients than

successful hoaxes, which implies that their outlinks are more coherent. It appears to be difficult to craft a fake concept that is embedded into the network of true concepts in a realistic way. In other words, making an article look realistic on the surface is easy; creating a realistic network fingerprint is hard.

As an aside, Fig. 4.6(d) is stratified by ego-network size because otherwise clustering coefficient and ego-network size could be confounded, as shown by the negative trend: when an article links to many other articles, they tend to be less tightly connected than when it links to only a few selected other articles—akin to a precision/recall tradeoff.

4.4.3 Support features

Something completely fabricated should never have been referred to before it was invented. Therefore we expect the frequency with which an article's name appears in other Wikipedia articles before it is created to be a good indicator of whether the article is a hoax.

Number of prior mentions. To test this hypothesis, we process Wikipedia's entire revision history (11 terabytes of uncompressed text) and, for each article A included in one of our four groups, identify all revisions from before A's creation time that contain A's title as a substring.

Of course, such a crude detector is bound to produce false positives.[5] But since the false-positive rate is likely to be similar across groups of articles, it is nonetheless useful for comparing different groups in a relative fashion, as done in Fig. 4.7(a), which shows that the two types of non-hoaxes (wrongly flagged and unflagged, *i.e.*, legitimate) have very similar distributions of prior mentions; analogously, the two types of hoaxes (successful and failed) resemble each other. One important difference between successful and failed hoaxes, however, is that of the successful

[5] For instance, a mention of the newspaper *The Times* will be spuriously detected in the Bob Dylan article because it mentions the song *The Times They Are a-Changin*.

ones, 40% are mentioned in at least one other article before creation, whereas this is the case for only 20% of the failed ones. (At 60% the rate is much higher for non-hoaxes.)

Time of first prior mention. Part of the reason why so many hoaxes have a mention before creation is due to the aforementioned false-positive rate of our simplistic mention detector. But there is a second reason: smart hoaxsters may carefully prepare the environment for the launch of their fabrication by planting spurious mentions in other articles, which creates an illusion of external support.

Consider Fig. 4.7(b), which plots the cumulative distribution function of the time between the appearance of the first mention of an article A in some other article and the creation of A itself. Legitimate articles are referred to long before they are created: 75% have been mentioned for over a year by the time the article is created, and under 5% have been mentioned for less than an hour. Successful hoaxes, on the contrary, have a probability of only 35% of having been mentioned for over a year when the hoax is created,[6] and a probability of 24% of having been mentioned for less than an hour—up by a factor of about 5 compared to non-hoaxes. We suspect that it is in many cases the hoaxster herself who inserts the first mention so briefly before creating the hoax in order to lend it artificial support.

Creator of first prior mention. Looking for additional evidence for this hypothesis, we explicitly investigate who is responsible for the first mention. To this end, Fig. 4.7(c) plots the fraction of first mentions made by the article creator herself. (Recall from Sec. 4.2 that we always know which user created an article, since anonymous users do not have permission to create new articles.) We expected most hoaxes to have been first mentioned by the hoaxster herself, but inspecting the

[6] This number is much larger for failed hoaxes, which begs an explanation. Eyeballing the data, we conjecture that this is caused by obvious, failed hoaxes often being created with mundane and commonplace names, such as “French immigrants” or “Texas style”.

figure we see that this is not the case: the fraction of first mentions inserted by the article creator is only slightly larger for hoaxes than for non-hoaxes (21% *vs.* 19%).

It seems that hoaxsters are smarter than that: Fig. 4.7(c) also tells us that 45% of first mentions are introduced by non-logged-in users identified only by their IP address, whereas the baseline over legitimate articles is only 19% here. Hence it seems likely that the anonymous user adding the first mention is often the hoaxster herself acting *incognito,* in order to leave no suspicious traces behind.

We conjecture that a significant fraction of first mentions from logged-in users other than the hoaxsters in fact stem from the hoaxsters, too, via fake "sockpuppet" accounts, but we have no means of verifying this hypothesis.

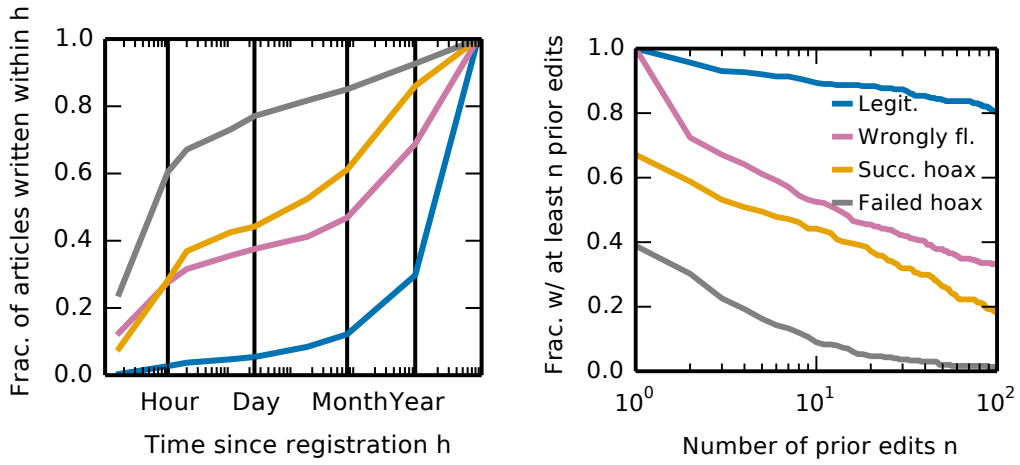
4.4.4 Editor features

The evidence from the last subsection that hoaxsters may act undercover to lend support to their fabricated articles motivates us to take a broader look at the edit histories of article creators.

Number of prior edits and editor age. Consider Fig. 4.8, where we explore article creators' edit histories under two metrics: the time gone by since the user registered on Wikipedia (Fig. 4.8(a)) and the number of edits they have made prior to creating the article in question (Fig. 4.8(b)).

The originators of typical legitimate articles are established members of the Wikipedia community: three-quarters of all such articles were started by editors who have been registered for more than a year, with a median of over 500 prior edits.[7] On the contrary, the three groups of articles that are flagged as hoaxes (whether they really are hoaxes or not) are created by much more recent accounts, in the following order: failed-hoax authors are the youngest members, followed

[7] In order to limit the number of calls to the Wikipedia API, we collected at most 500 edits per user. Therefore, the median measured in this setting (500) is a lower bound of the real median.



(a) Time since registration

(b) Number of prior edits

Figure 4.8: Editor features: (a) CDF of time between account registration and article creation. (b) CCDF of number of edits by same user before article creation.

by the creators of successful hoaxes, and finally by those of articles flagged wrongly as hoaxes.

In particular, while only about 3% of legitimate-article authors create the article within the hour of registration, the fractions are 60% for creators of failed hoaxes, and 25% for those of successful hoaxes and wrongly flagged articles. In the case of wrongly flagged articles, we suspect that inexperience may cause users to write articles that do not comply with Wikipedia's standards (*cf.* Fig. 4.5). This in combination with the concern that, due to the recent registration date, the account might have been created specifically for creating the hoax might lead patrollers to erroneously suspect the new article of having been fabricated.

4.5 Experimental Evaluation: Automatic hoax detection

Having gained several valuable insights on the characteristics of Wikipedia hoaxes and their differences from other types of articles, we are now in a position to apply these findings by building machine-learned classifiers to automate some important decisions revolving around hoaxes. We

consider the following four tasks:

1. Will a hoax get past patrol?
2. How long will a hoax survive?
3. Is an article a hoax?
4. Is an article flagged as such really a hoax?

The first two tasks take the hoaxster's perspective and ask how high the chances are of the hoax being successful. The latter two tasks take the patrollers' perspective and aim to help them make an accurate decision during patrol and after.

All classifiers use the same algorithm and features, but are fitted on different training sets of positive and negative examples. This allows us to analyze the fitted weights in order to understand what features matter most in each task.

Classification algorithm. We experimented with a variety of classification algorithms—logistic regression, support vector machines, and random forests—and found the latter to work best. Hence all results reported here were obtained using random forests [131].

We use balanced training and test sets containing equal numbers of positive and negative examples, so random guessing results in an accuracy, as well as an area under the receiver operating characteristic (ROC) curve (AUC) of 50%.

Features. All features used by the classifier have been discussed in detail in Sec. 4.4 and are summarized in Table 4.2.

In the rest of this section we provide more details on each of the four tasks (Sec. 4.5.1) and then move on to presenting and discussing the results we obtained (Sec. 4.5.2).

Feature	Group
Plain-text length	Appearance (Sec. 4.4.1)
Plain-text-to-markup ratio	Appearance
Wiki-link density	Appearance
Web-link density	Appearance
Ego-network clustering coefficient	Network (Sec. 4.4.2)
Number of prior mentions	Support (Sec. 4.4.3)
Time of first prior mention	Support
Creator of first prior mention	Support
Number of prior edits	Editor (Sec. 4.4.4)
Editor age	Editor

Table 4.2: Features used in the random-forest classifiers.

4.5.1 Classification tasks

Task 1: Will a hoax get past patrol? Here the objective is to predict if a hoax will pass the first hurdle in its life cycle (Fig. 4.1), *i.e.*, if it will manage to trick the patroller into believing that it is a legitimate article.

Such a classifier could tell the hoaxster whether the hoax is ready to be submitted to the patrolling process yet. It would also be useful from the patroller's perspective because the fitted feature weights can give us insights into which features make a hoax slip through patrol; we could then counteract by scrutinizing those characteristics more carefully.

Here the set of positive examples consists of all 2,692 hoaxes that were not flagged by the users who patrolled them. The negative examples are sampled randomly from the set of 12,901 hoaxes that are correctly flagged by the patroller, while ensuring that for each positive article we have a negative article created on the same day.

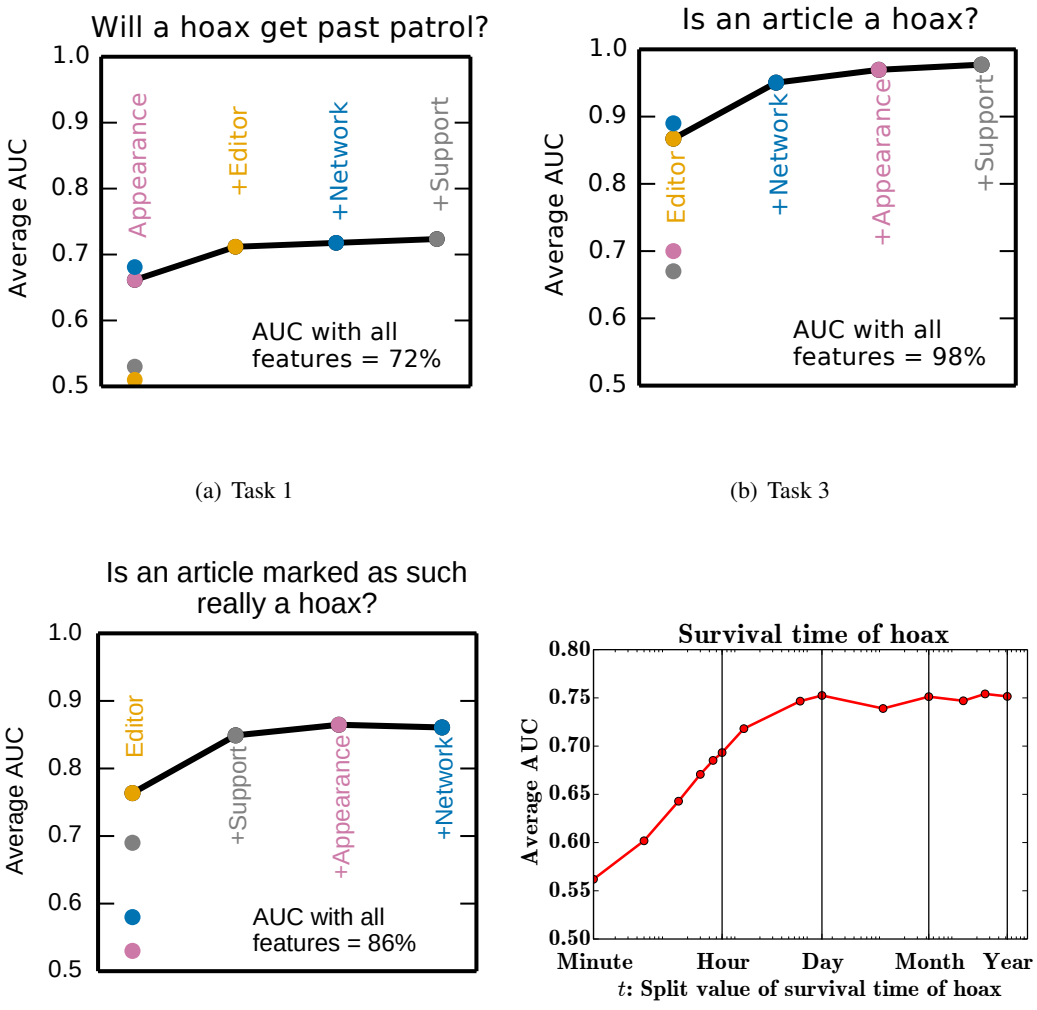
Task 2: How long will a hoax survive? Our second task is to predict the survival time of hoaxes that have managed to pass patrol, defined as the time between patrol and flagging (Fig. 4.1). We phrase this as a binary decision problem by fixing a threshold τ and asking whether a hoax will survive for at least τ minutes. We repeat this task for various values of τ, ranging from one minute to one year.

Given τ, the positive examples are all hoaxes that survived for at least τ minutes from patrol to flagging. The negative set consists of hoaxes flagged within τ minutes from patrol. The larger of the two sets for the given τ is subsampled to match the smaller set in size.

Task 3: Is an article a hoax? In this task, the classifier is supposed to assess if an article that has passed patrol is a hoax or not. In the language of Fig. 4.1, the task aims to automate the flagging step. This classifier could be employed to double-check the decisions made by human patrollers and thereby decrease their false-negative rate.

Here the positive examples are the 2,692 articles that passed patrol but were later flagged as hoaxes and deleted. As argued in the introduction, the most detrimental hoaxes are those that survive for a long time and attract significant traffic. In order to equip our classifier with the ability to detect this subclass, we include only those 301 hoaxes as positive examples that have existed for at least 30 days from creation to flagging and that have received an average of at least 5 pageviews during this time. For each hoax in the positive set we randomly sample one negative example from among all articles that were created on the same day as the hoax and were never flagged or deleted.

Task 4: Is an article marked as such really a hoax? The final classification task deals with the scenario in which an article has been flagged as a hoax by a Wikipedia user, and our goal is to double-check if the article is indeed a hoax. That is, this classifier is supposed to act as a safeguard between the flagging and deletion steps (Fig. 4.1).



(a) Task 1

(b) Task 3

(c) Task 4

(d) Task 2

Figure 4.9: (a–c) Results of forward feature selection for tasks 1, 3, 4. (d) Performance (AUC) on task 2 as function of threshold τ.

In other words, while task 3 aims to decrease human patrollers' false-negative rate, the classifier developed here may decrease their false-positive rate. This could be very valuable because false positives come at a large cost: if an article is unjustly deleted as a hoax, this might discourage the editor to contribute further to Wikipedia.

The negative set comprises the 960 articles that were wrongly flagged, *i.e.*, that were later acquitted by having the hoax flag removed and were never deleted. Candidates for positive examples are all articles that were flagged as hoaxes and eventually deleted. To create a balanced dataset, we pair each negative example with a positive examples whose creation and flagging dates are closely aligned with those of the negative example (we use propensity score matching [132] to perform the pairing).

4.5.2 Results

Table 4.3 reports the performance on tasks 1, 3, and 4 when using all features of Table 4.2. Task 2 depends on the threshold τ, so we plot the AUC as function of τ in Fig. 4.9(d).

	Task	**Acc.**	**AUC**
1	Will a hoax get past patrol?	66%	71%
3	Is an article a hoax?	92%	98%
4	Is an article flagged as such really a hoax?	76%	86%

Table 4.3: Classification results; for task 2, *cf.* Fig. 4.9(d).

Maybe surprisingly, deciding if an article is a hoax or not (task 3) is the easiest task, with an accuracy (AUC) of 92% (98%). Performance is also quite high on the task of deciding whether something that has been flagged as a hoax is really one (task 4); here we achieve an accuracy (AUC) of 76% (86%). The hardest tasks are to predict if a hoax will pass patrol (task 1; accuracy 66%, AUC 71%) and how long it will survive once it has passed patrol (task 2): Fig. 4.9(d) shows

that the AUC increases with the threshold τ, but levels off at 75% around $\tau = 1$ day. That is, one day seems to be a natural threshold that separates successful from failed hoaxes. This echoes our finding from Fig. 4.2(b), where we saw that surviving the first day immensely boosts the odds of surviving for longer.

Feature importance. In order to understand which features are important for which task, we evaluate smaller models that consist of only one of the four feature groups (Table 4.2). The performance of these smaller models is shown by the vertically aligned dots in the leftmost columns of Fig. 4.9(a)–4.9(c). For tasks 3 and 4, which deal with deciding if something is a hoax, features of the creator's edit history are most effective; on task 3 (hoax *vs.* non-hoax), the network feature (ego-network clustering coefficient) does equally well. Task 1, where we predict if a given hoax will pass patrol, profits most from appearance and editor features.

Next, we perform forward feature selection to understand what the marginal values of additional features are. The results are plotted as the black curves in Fig. 4.9(a)–4.9(c).[8] The conclusion is that all feature groups contribute their share, but with diminishing returns.

Trawling Wikipedia for hoaxes. In order to find hoaxes that are still present in Wikipedia, we deployed the hoax-*vs.*-non-hoax classifier on Wikipedia's entire revision history. We discuss the results in detail online.[9] To give but two examples, our algorithm identified the article about "Steve Moertel", an alleged Cairo-born U.S. popcorn entrepreneur, as a hoax. The article was deleted by an editor who confirmed the article's hoax status after we had flagged it—and after it had survived in Wikipedia for 6 years and 11 months. Similarly, we flagged the article about "Maurice Foxell", an alleged children's book author and Knight Commander of the Royal Victorian Order; the article

[8] We performed forward feature selection on the training set and report performance on the testing set. This is why the first selected feature may have lower performance than other features.

[9] http://snap.stanford.edu/hoax/

was deleted by an editor after it had survived for 1 year and 7 months.

4.6 Human guessing experiment

The observational analysis of Sec. 4.4 allowed us to gain many insights, but it also has some shortcomings. First, survival time defined by the period between patrol and flagging is not a perfect indicator of the quality of a hoax, as the hoax may have survived for a long time for a variety of reasons; *e.g.*, it may be the case that the false information is disguised in a truly skillful manner, or simply that it was sloppily patrolled and was afterwards seen by only few readers who could have become suspicious. So by only considering the observational data we have analyzed above, we cannot know which hoax survived for which reason.

Second, the binary label whether a hoax passed patrol or not is not necessarily representative of how likely a regular Wikipedia reader, rather than a patroller, would be to believe the hoax. Patrollers are encouraged to base their decision on all available information, including fact-checking on the Web via search engines, verifying included references, inspecting the article creator's edit history, etc. We suspect that most Wikipedia readers do not use such devices during casual reading and are therefore more likely to fall prey to a hoax that looks legitimate on the surface.

To overcome these shortcomings and understand what makes a hoax credible to average readers rather than patrollers, we now complement our observational findings with an experiment. The idea is to (1) create an identical situation of scrutiny across a variety of hoaxes, thus mitigating the first concern from above, and (2) disallow the use of external resources such as search engines, thus addressing the second concern.

4.6.1 Methodology

In designing the experiment, we start by selecting 64 successful hoaxes according to the definition from the beginning of Sec. 4.4. We then create an equally sized set of legitimate, non-hoax articles such that (1) for each hoax we have a legitimate article created on the same day as the hoax and (2) the two sets have nearly identical distributions of the appearance features of Sec. 4.4.1, which we achieve via propensity score matching [132].[10]

We then created 320 random hoax/non-hoax pairs such that each hoax was paired with 5 distinct non-hoaxes and *vice versa*. These pairs were then shown side-by-side in random order to human raters on Amazon Mechanical Turk, who were asked to decide which of the two articles is a hoax by only looking at the text and not searching the Web. Each pair was given to 10 raters, so we collected 3,200 labels in total (50 per hoax).

4.6.1.1 Quality assurance in human guessing experiment

Hoax/non-hoax pairs were issued in batches of 3; one of the 3 pairs was a test pair for which we made sure it was obvious which article was legitimate, by choosing an article about a country as the non-hoax. Raters were told they would not be paid if they did not get the test pair right (they were not told which one it was). This was to incentivize raters to make a best effort on all 3 pairs and refrain from clicking randomly. It also allows us to discard answers from raters who answered less than a minimum fraction of test pairs correctly. 92% of the test questions were answered correctly, and we discard all answers from raters with a test-question accuracy below 75%, which leaves us with 2,942 of the original 3,200 pairs.

[10] We additionally balance the sets with respect to the numbers of sections, images, and references in the articles.

4.6.2 Results

Human *vs.* classifier accuracy. Human accuracy on all rated pairs is 66%. The macro-average that gives equal weight to all users (hoaxes) is 63% (66%). Given that random guessing on the task would give 50%, this performance is surprisingly weak.[11] In comparison, we tested our hoax-*vs.*-non-hoax classifier (task 3 of Sec. 4.5) on the same pairs shown to humans and achieved an accuracy of 86%, thus outperforming humans by a large margin.[12]

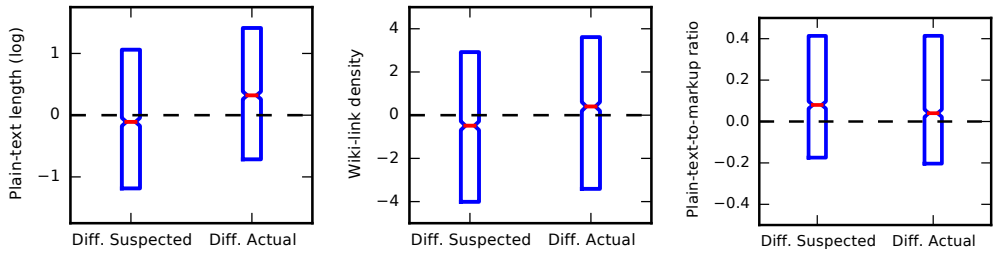
This classifier used all features of Sec. 4.5. The human, however, saw only the articles themselves and was not allowed (and for most features not even able to) take network, support, and editor features into account. To allow for a fairer comparison, we therefore also tested a version of our classifier that uses only appearance features, obtaining an accuracy of only 47%. This weak (roughly random) performance is to be expected, since the sets of hoaxes and non-hoaxes were constructed to have very similar distributions with respect to appearance features (*cf.* above), so these features should be uninformative for the task.

We conclude that features that look beyond the surface, such as the article creator's edit history, the mentions received from other articles, and the density of the article's ego network, are of crucial importance for deciding whether an article is a hoax: they make the difference between random and above-human performance.

Human bias. Our next goal is to understand what factors humans go by when deciding what is a hoax. We proceed as follows: given a feature f of interest (such as plain-text length), compute

[11] One might object that humans possibly *did* guess randomly, but we guarded against this via the quality-assurance mechanism described in Section 4.6.1.1.

[12] Since testing is done on pairs, we also trained the classifier on pairs: as the feature vector for a pair, we use the difference of the feature vectors of the left and right articles, and the classifier is tasked to predict whether the left or right article is the hoax. The training pairs did not contain articles appearing in the test pairs.

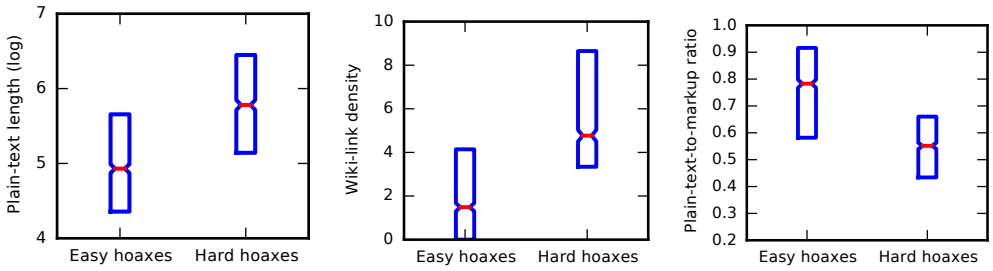


(a) Plain-text length (b) Wiki-link dens. (c) Plain-text/markup

Figure 4.10: Human bias in the guessing experiment with respect to three appearance features f. Left boxes: difference δ of suspected hoax minus suspected non-hoax. Right boxes: difference δ^* of actual hoax minus actual non-hoax.

the within-pair difference δ of the *suspected* hoax minus the suspected non-hoax for each pair. Similarly, compute the difference δ^* of the *actual* hoax minus the actual non-hoax, and compare the distributions of δ and δ^*. Now, if δ tends to be lower than δ^*, this implies that humans tend to think that lower values of f indicate hoaxes, although they would have had to choose the higher values more frequently in order to guess perfectly; in other words, they are biased to believe that articles with lower values of f are hoaxes.

Our findings from this analysis are displayed in the boxplots of Fig. 4.10. Here, the left box of each subfigure summarizes the distribution of δ, and the right box, that of δ^*. For instance, Fig. 4.10(a) shows that the suspected hoax tends to be shorter than the suspected non-hoax, whereas the actual hoax tends to be longer than the actual non-hoax. So humans have a bias towards suspecting short articles to be hoaxes that is not warranted by the dataset at hand. Similarly, we find that humans are led to believe that articles with a lower wiki-link density (Fig. 4.10(b)) and, to a lesser extent, with a higher plain-text-to-markup ratio (*i.e.*, less wiki markup; Fig. 4.10(c)), are hoaxes. Flipped around, from the hoaxster's perspective this means that a hoax stands a higher chance of succeeding if it is longer and looks more like a typical Wikipedia article.



(a) Plain-text length (b) Wiki-link dens. (c) Plain-text/markup

Figure 4.11: Comparison of easy- and hard-to-identify hoaxes with respect to three appearance features.

Next we create two groups of hoaxes: those that are easy, and those that are hard, to detect for humans. To define these groups we first rank all hoaxes in increasing order according to the probability with which humans identified them correctly; the upper third then defines the easy, and the lower third the hard, cases. For each feature we then compare the distributions within the two groups. The results, shown in Fig. 4.11, indicate that the log median number of plain-text words of the hard group is higher by about 1 than that for the easy group, *i.e.*, hard-to-recognize hoaxes are in the (non-log) median about $e^1 \approx 2.7$ times as long as easy-to-recognize hoaxes. Similarly, hoaxes with many wiki links (Fig. 4.11(b)) and a low plain-text-to-markup ratio (Fig. 4.10(c)), *i.e.*, with many wiki-specific elements, are difficult to recognize.

Examples. Of course, it is not only simple structural and superficial features such as the length, link density, and presence of wiki-specific elements that determine if an article is recognized as a hoax. It is also, and to a large extent, the semantic content of the information conveyed that matters. Therefore we conclude our discussion of the human experiment with some qualitative remarks. Table 4.4 lists the hardest (top) and easiest (bottom) hoaxes (left) and non-hoaxes (right) for humans to identify correctly, where "hardness" is captured by the fraction of humans who failed to identify the article correctly across all pairs it appeared in. Hard-to-identify hoaxes are

Acc.	Hoax	Acc.	Non-hoax
0.333	TV5 (Malaysia)	0.292	Titeica
0.341	Tom Prescillo	0.312	DXMM
0.362	Alexander Ivanovich	0.364	Better Made Potato Chips Inc.
	Popov	0.370	Olympiacos B.C. vs Punch Delft
0.391	Noah Chazzman		(prehistory)
0.400	Dav Sorado	0.378	Don't Come Home for Christmas
...	...	...	...
0.867	The Oregon Song	0.872	List of governors of Islamic Egypt
0.875	Nicktoons: Dark Snap	0.891	Bobby Brown discography
0.884	Breast Touching Festival	0.907	List of Naruto episodes (season 4)
	of China	0.957	Alpine skiing at the 2002 Winter
0.955	Burger King Stunners		Olympics – Women's slalom
0.957	Mama Mo Yeah	0.958	USS Charles P. Crawford (SP-366)

Table 4.4: Hoaxes (left) and non-hoaxes (right) that were hardest (top) and easiest (bottom) for humans to identify correctly.

often elaborate articles about fake people, whereas the easy ones are oftentimes already given away by their titles.

The non-hoaxes that were least credible to raters frequently have titles that sound tongue-in-cheek. The article on the (real) Philippine radio station DXMM might have been mistaken so often because the version used in the experiment was very short and had no wiki links and sections, or because it was clumsily phrased, calling the station "the fruit of missions made by the Missionary Oblates of Mary Immaculate in the difficult and harsh fields of Mindanao and Sulu archipelago in southern Philippines."

4.7 Conclusion

In this chapter we investigated impact, characteristics, and detection of hoax articles on Wikipedia. We utilized a rich labeled dataset of previously discovered hoaxes and use it to assess the real-world impact of hoax articles by measuring how long they survive before being debunked, how many pageviews they receive, and how heavily they are referred to by documents on the Web. We found that the Wikipedia community is efficient at identifying hoax articles, but that there is also a small number of carefully crafted hoaxes that survive for a long time and are well cited across the Web.

We also characterized successful hoaxes by comparing them with legitimate articles and with failed hoaxes that were discovered shortly after being created. We uncovered characteristic differences in terms of article structure and content, embeddedness into the rest of Wikipedia, and features of the editor who created the hoax.

We rely on these lessons to build an automatic classification system to determine whether a given article is a hoax. By combining features derived from the article's appearance, its mentions in other articles, and its creator, as well as the Wikipedia hyperlink network, our approach achieved an AUC/ROC of 98%. We also compared our automatic hoax detection tool with the performance of human evaluators and find that humans without any specialized tools are not skilled at discerning hoaxes from non-hoaxes (63% accuracy). Our experiments showed that, while humans have the tendency to rely on article appearance features, those alone are not sufficient to make accurate judgments. In contrast, our algorithms were able to utilize additional signals, such as the embeddedness of the article into the rest of Wikipedia, as well as properties of the article creator, in order to accurately identify hoaxes. To turn our insights into actions, we applied our learned model to Wikipedia's entire revision history and find hoaxes that have been hidden in it for a long

time.

Chapter 5

Detecting sockpuppets in online discussion communities

The anonymity afforded by web platforms often leads to some users deceiving others using multiple accounts, or *sockpuppets* [15], in order to manipulate public opinion [32, 33] and vandalize content (e.g., on Wikipedia [34]). In this chapter, we focus on identifying, characterizing, and predicting sockpuppetry by studying *nine online discussion communities* [35].

5.1 Introduction

Discussions are a core mechanism through which people interact with one another and exchange information, ideas, and opinions. They take place on social networks such as Facebook, news aggregation sites such as Reddit, as well as news websites such as CNN.com. Nonetheless, the anonymity afforded by some discussion platforms has led to some users deceiving others using multiple accounts, or sockpuppets [15]. Sockpuppetry is often malicious and deceptive, and has been used to manipulate public opinion [32, 33] and vandalize content (e.g., on Wikipedia [34]).

Prior work on sockpuppetry and deceptive behavior has tended to focus on individual mo-

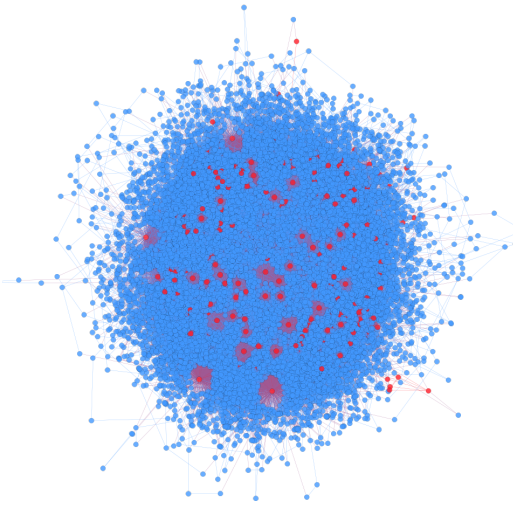


Figure 5.1: AVClub.com social network. Nodes represent users and edges connect users that reply to each other. Sockpuppets (red nodes) tend to interact with other sockpuppets, and are more central in the network than ordinary users (blue nodes).

tivations [60, 133], or on identifying sockpuppets through their linguistic traits [134] (e.g., on Wikipedia [34, 37]). Further, given the difficulty of obtaining ground-truth data about sockpuppets, work has also tended to make assumptions about how sockpuppets behave, for example, assuming that they have similar usernames [135], they are only used to support one another [136], or that they write similar to each other [134]. Further, research has generally not considered how the interactions between sockpuppets controlled by the same individual could be used to accurately and automatically identify sockpuppets. As such, improved methods for identifying sockpuppets, as well as deeper analyses of how sockpuppets interact with one another may allow us to better understand, characterize, and automatically detect sockpuppetry.

In this chapter, we focus on identifying, characterizing, and predicting sockpuppetry in nine different online discussion communities. We broadly define a *sockpuppet* as a user account that is controlled by an individual (or *puppetmaster*) who controls at least one other user account. By considering less easily manipulated behavioral traces such as IP addresses and user session

data, we automatically identified 3,656 sockpuppets comprising 1,623 *sockpuppet groups*, where a group of sockpuppets is controlled by a single puppetmaster.

Studying these identified sockpuppets, we discover that sockpuppets differ from ordinary users in terms of how they write and interact with other sockpuppets. Sockpuppets have unique linguistic traits, for example, using more singular first-person pronouns (e.g., "I"), corroborating with prior work on deception [32]. They also use fewer negation words, perhaps in an attempt to appear more impartial, as well as fewer standard English parts-of-speech such as verbs and conjunctions. Suggesting that sockpuppets write worse than ordinary users on average, we find that posts are more likely to be downvoted, reported by the community, and deleted by moderators. Sockpuppets also start fewer discussions.

Examining pairs of sockpuppets controlled by the same puppetmaster, we find that sockpuppets are more likely to post at the same time and post in the same discussion than random pairs of ordinary users. As illustrated in Figure 5.1, by studying the network of user replies, we find that sockpuppets have a higher pagerank and higher local clustering coefficient than ordinary users, suggesting that they are more important in the network and tend to generate more communication between their neighbors. Further, we find that pairs of sockpuppets write more similarly to each other than to ordinary users, suggesting that puppetmasters tend not to have both "good" and "bad" accounts.

While prior work characterizes sockpuppetry as malicious [34, 36, 37], not all the sockpuppets we identified were malicious. In some sockpuppet groups, sockpuppets have display names significantly different from each other, but in other groups, they have more similar display names. Our findings suggest a dichotomy in how deceptive sockpuppets are – some are *pretenders*, that masquerade as separate users, while others are *non-pretenders*, that is sockpuppets that are overtly visible to other members of the community. Pretenders tend to post in the same discussions and

are more likely to have their posts downvoted, reported, or deleted compared to non-pretenders. In contrast, non-pretenders tend to post in separate discussions, and write posts that are longer and more readable.

Our analyses also suggest that sockpuppets may differ in their supportiveness of each other. Pairs of sockpuppets controlled by the same puppetmaster differ in whether they agree with each other in a discussion. While sockpuppets in a pair mostly remain neutral towards each other (or are *non-supporters*), 30% of the time, one sockpuppet in a pair is used to support the other (or is a *supporter*), while 10% of the time, one sockpuppet is used to attack the other (or is a *dissenter*). Studying both deceptiveness and supportiveness, we find that supporters also tend to be pretenders, but dissenters are not more likely to be pretenders, suggesting that deceptiveness is only important when sockpuppets are trying to create an illusion of public consensus.

Finally, we show how our previous observations can be used to develop models for automatically identifying sockpuppetry. We demonstrate robust performance in differentiating pairs of sockpuppets from pairs of ordinary users (ROC AUC=0.90), as well as in the more difficult task of predicting whether an individual user account is a sockpuppet (ROC AUC=0.68). We discover that the strongest predictors of sockpuppetry relate to interactions between the two sockpuppet accounts, as well as the interactions between the sockpuppet and the community.

Altogether, our results begin to shed light on how sockpuppetry occurs in practice, and paves the way towards the development and maintenance of healthier online communities.

5.2 Data and Definitions

We start by laying out the terminology that we use in the remainder of this chapter. We then describe the data we used in our analysis and a robust method for automatically identifying sockpuppets.

Sockpuppetry. While in prior work sockpuppets have typically been used to refer to a false online identity that is used for the purposes of deceiving others [135, 136], we adopt a broader definition. We define a *sockpuppet* as a user account controlled by an individual who has at least one other account. In other words, if an individual controls multiple user accounts, each account is a sockpuppet. The individual who controls these sockpuppets is referred to as the *puppetmaster*. We use the term *sockpuppet group/pair* to refer to all the sockpuppets controlled by a single puppetmaster. In each sockpuppet group, one sockpuppet typically has made significantly more comments than the others – we refer to this sockpuppet as the *primary* sockpuppet, and the other sockpuppets as *secondary*. Finally, we use *ordinary user* to refer to any user account that is not a sockpuppet.

We study sockpuppets in the context of online discussion communities. In these communities, people can create accounts to comment on articles. In addition to writing or replying to *posts*, users can also vote on posts or report them for abuse. Moderators can, in turn, delete posts that do not conform to community standards. If a post is not a reply to another post, we call that post a *root post*. We define a *discussion* as all the posts that follow a given news article, and a *sub-discussion* as a root post and any replies to that post.

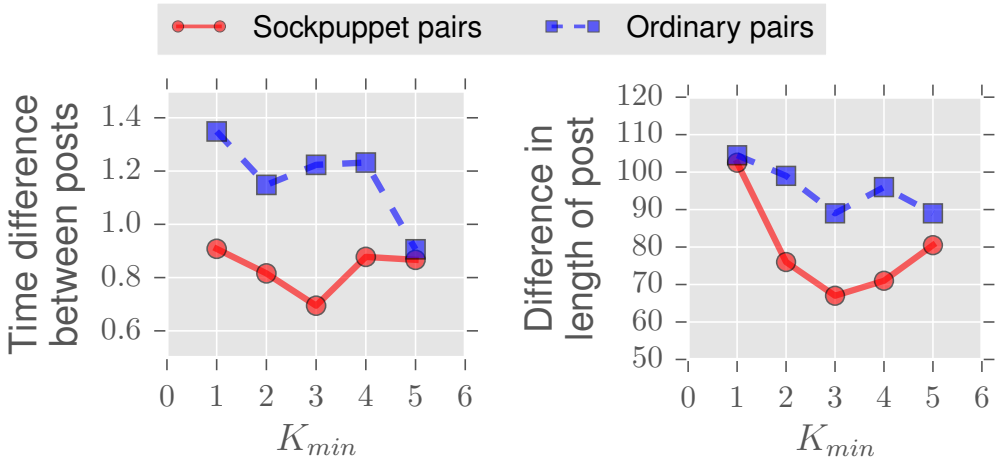
Data. The data consists of nine different online discussion communities that encompass a variety of topical interests – from news and politics to sports and entertainment (Table 5.1). Disqus, a commenting platform that hosted these discussions, provided us with a complete trace of user activity across nine communities that consisted of 2,897,847 users, 2,129,355 discussions, and 62,744,175 posts. Each user has a display name which appears next to their posts and an email address which is private. (To respect user privacy, all email addresses were stripped of the domain names and analyzed in aggregate.) Each post is also associated with an anonymized IP address of the posting user.

Community	Genre	# Users	# Sockpuppets	# Sock-groups
CNN	News	846,436	1,191	523
Breitbart	News	196,846	761	352
allkpop	Music	159,671	445	193
MLB	Sports	115,845	339	139
IGN	Games	266,976	314	142
Fox News	News	145,009	214	94
A.V. Club	Entertainment	37,332	199	90
The Hill	Politics	158,378	134	62
NPR	News	65,662	59	28

Table 5.1: Statistics of the nine online discussion communities.

Identifying sockpuppets. No explicit labels of sockpuppets exist in any of the discussion communities, so to identify sockpuppets, we use multiple signals that together suggest that accounts are likely to share the same owner – the IP address of a comment, as well as the times at which comments are made, and the discussions they post in. Our approach draws on the approach adopted by Wikipedia administrators who identify sockpuppets by finding accounts that make similar edits on the same Wikipedia article, in near-similar time and from same IP address [137]. As we are primarily interested in identifying sockpuppets with high precision, the criteria we adopt is relatively conservative – relaxing these criteria may improve recall, but at the cost of more false positives.

To limit spurious detection of sockpuppets, we filter the data and remove any IP addresses used by many user accounts, as these accounts may simply be accessed from behind a country-wide proxy or intranet. We also do not consider user accounts that post from many different IP addresses, since they have high chance of sharing IP address with other accounts. Specifically, for each discussion community, we remove the top 5% most used IP addresses and the top 5% accounts that have the most IP addresses.



(a) (b)

Figure 5.2: Varying K_{min}, the minimum number of overlapping sessions between users for them to be identified as sockpuppets. For sockpuppet pairs (blue) the time between posts, and the difference in post lengths reach a minimum value at $K_{min} = 3$.

Then, we identify sockpuppets as user accounts that post from the same IP address in the same discussion within T minutes for at least K_{min} different discussions. Here, we set $T =$ 15 minutes (larger values result in empirically similar findings). To pick an appropriate value for K_{min}, we use two metrics that prior work has found indicative of sockpuppets: the time difference between posts made by two accounts, and the difference in the length of posts [37, 134, 138, 139].

Figure 5.2 plots these quantities for identified pairs of sockpuppets as well as random pairs of users. We observe that regardless of the value of K_{min}, the identified sockpuppets post more closely in time and write posts more similar in length, compared to a random pair of users. Further, we find that among sockpuppet pairs, these quantities achieve their minimum at $K_{min} = 3$, which means that sockpuppets are most reliably identified at that value of K_{min}.

To summarize, we define sockpuppets as user accounts that post from the same IP address in the same discussion in close temporal proximity at least 3 times. We then define sockpuppet

groups as maximal sets of accounts such that each account satisfies the above definition with at least one other account in the group. Overall, we identify a total of 1,623 sockpuppet groups, consisting of 3,656 sockpuppets from nine different online discussion communities (Table 5.1). As most sockpuppet groups contain two sockpuppets (Figure 5.3(a)), we focus our analyses on pairs of sockpuppets.

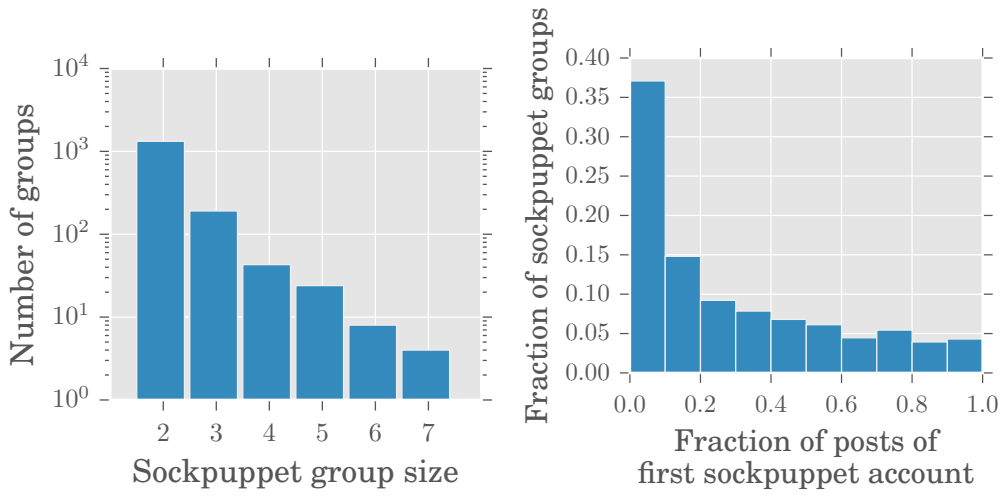
We give an example of an identified sockpuppet group below, which consists of three users: S_1 and S_2 comprise a pair of sockpuppets, while O is an ordinary user. After an unusually positive interaction between the two sockpuppets, O identifies them as being controlled by the same puppetmaster:

> S_1: Possibly the best blog I've ever read major props to you.
>
> $\hookrightarrow S_2$: Thanks. I knew Marvel fans would try to flame me, but they have nothing other than "oh that's your opinion" instead of coming up with their own argument.
>
> $\hookrightarrow O$: Quit talking to yourself, *******. Get back on your meds if you're going to do that.

5.3 Characterizing Sockpuppetry

Having identified sockpuppets, we now turn to characterizing their behavior. We study when sockpuppets are created and how their language and social networks differ from ordinary users across all nine discussion communities.

Sockpuppets are created early. To understand when sockpuppets are created, we examine the activity of the first sockpuppet account in each sockpuppet pair. Figure 5.3(b) shows the fraction of total number of posts made by the first sockpuppet before the second sockpuppet is created. The second sockpuppet tends to be created during the first 10% of the posts, with a median of 18%



(a) (b)

Figure 5.3: (a) Number of sockpuppet groups, *i.e.* sockpuppets belonging to the same puppetmaster. (b) The second sockpuppet in a group tends to be created shortly after the first.

posts written by first sockpuppet before the second sockpuppet begins posting. In other words, sockpuppets tend to be created early in a user's lifetime, which may indicate that sockpuppet creation is premeditated and not a result of user's interactions in the community.

Matching sockpuppets with ordinary users. On average, sockpuppets write more posts than ordinary users (699 vs. 19) and participate in more discussions (141 vs. 7). To control for this disparity, in all our subsequent analyses we use propensity score matching [132] to match sockpuppets with ordinary users that have similar numbers of posts and make posts to the same set of discussions.

5.3.1 Do puppetmasters lead double lives?

First, we explore an important question about how the behavior of sockpuppets S_1 and S_2 controlled by the same puppetmaster relates to that of an ordinary user O. Two possible hypotheses are illustrated in Figure 5.4. First is the *double life* hypothesis, where the puppetmaster maintains a

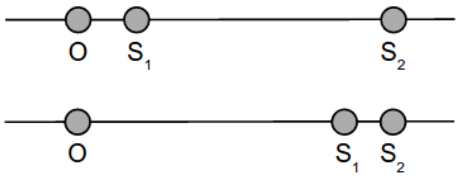


Figure 5.4: Two hypotheses how similarity of sockpuppet pairs and ordinary users relates to each other. Top: Under the double life hypothesis, sockpuppet S_1 is similar to an ordinary user O, while S_2 deviates. Bottom: Alternative hypothesis is that both sockpuppet accounts are highly different from ordinary users.

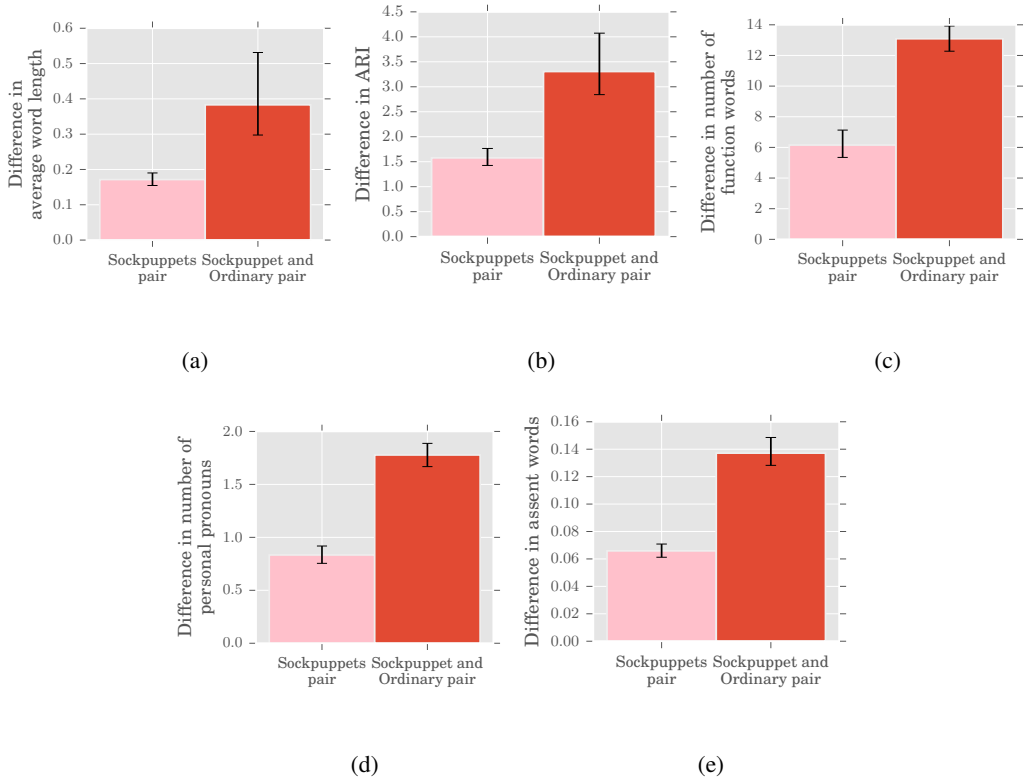


Figure 5.5: Difference in properties of sockpuppet pairs and that of sockpuppet-ordinary pairs. Sockpuppet pairs are more similar to each other in several linguistic attributes.

distinct personality for each sockpuppet – one sockpuppet, S_1, behaves like an ordinary user, while the other, S_2 behaves more maliciously. Under this hypothesis we would expect that the linguistic similarity of posts written by O and S_1 would be high, and that between both S_1 and S_2, as well

as O and S_2 to be significantly lower. In the alternative hypothesis (Figure 5.4 (bottom)), both sockpuppets act maliciously. In this case, we might expect that the linguistic similarity between S_1 and S_2 would be low, but that between S_1 and O, and S_2 and O to be much lower.

To find out which is the case, we compare the language of pairs of sockpuppets, and that of each sockpuppet with an ordinary user. To control for user activity, we again match sockpuppets with ordinary users that have similar posting activity, and that participate in similar discussions. Specifically, for each user, we created a feature vector consisting of several linguistic features computed from that user's posts, including LIWC categories and sentiment, the average number of words in a post, the average fraction of special characters. We then compute the cosine similarity of the feature vectors.

We find that on average, the two sockpuppets are more similar to each other than either is to an ordinary user ($p < 0.001$). Figure 5.5 highlights that these observations hold for individual features as well – the difference between two sockpuppets' readability score (ARI), average word length, number of function words, personal pronouns and assent words are smaller than that of either sockpuppet and an ordinary user. This suggests that the double life hypothesis (Figure 5.4(top)) is less likely to be true than the alternate hypothesis (Figure 5.4(bottom)).

In other words, this experiment suggests that puppetmasters do not lead double lives, and that it is generally not the case that individual sockpuppets controlled by the same puppetmaster behave differently. Rather, sockpuppets as a whole tend to write differently from ordinary users, and sockpuppets controlled by the same puppetmaster all tend to write similarly to each other.

5.3.2 Linguistic Traits of Sockpuppets

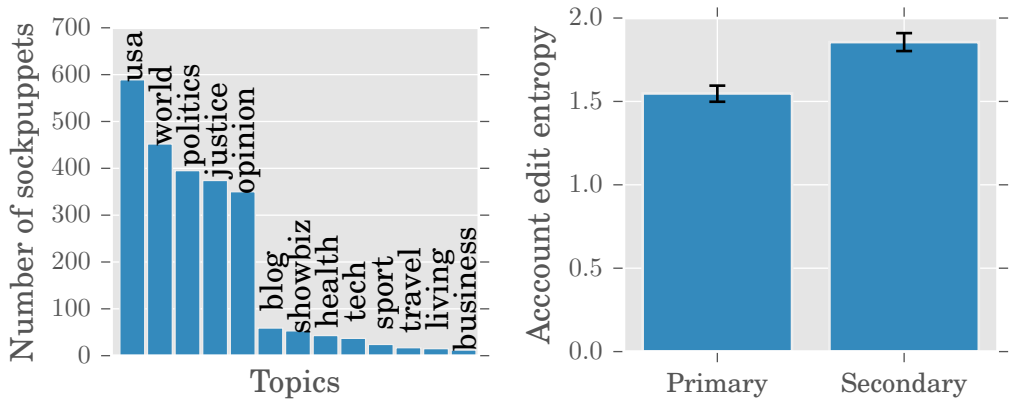
Having established that different sockpuppets controlled by the same puppetmaster behave consistently, we now turn our attention to quantify their linguistic traits more precisely. Here, we

focus on comparing various measures of similarity $sim(S_i, O)$ of a sockpuppet S_i ($i = \{1, 2\}$) and a matched ordinary user O. Specifically, we use LIWC word categories [140] to measure the fraction of each type of words written in all posts, and VADER [141] to measure sentiment of posts. We report the average values for sockpuppets and the corresponding p-values by performing paired t-tests for each sockpuppet and its matching ordinary user.

Do sockpuppets write differently from ordinary users? Linguistic traits have been used to identify trolls in discussions [78], vandalism on Wikipedia [142], and fake reviewers on e-commerce platforms [55]. For example, deceptive authors tend to increase their usage of function words, particles and personal pronouns [32, 74]. They use more first- and second-person singular personal pronouns (e.g., 'I', 'you'), while reducing their use of third-person singular personal pronouns (e.g., 'his', 'her'). They also tend to oversimplify their writing style by writing shorter sentences and writing words with fewer syllables.

We make similar observations with respect to sockpuppets. First, we observe that they tend to write posts that are more self-centered – and use "I" more often than ordinary users (0.076 for sockpuppets vs 0.074 for ordinary users, p <0.001). Sockpuppets also use "you" more often (0.017 vs 0.015, p <0.01) but third-person singular personal pronouns and plural personal pronouns (i.e., 'we', 'he', 'she', and 'they') less (*e.g.* 0.016 vs 0.018 for 'he/she' words, $p <$ 0.001), indicating that they tend to address other users in the community more directly. Similarly, we observe that sockpuppets also write shorter sentences than ordinary users (a mean of 12.4 vs. 12.9 words per sentence, $p < 0.001$). However, in contrast to prior work on deceptive writing, sockpuppets use a similar number of syllables per word (1.29 vs 1.28, $p = 0.35$).

Turning to differences in LIWC categories, we observe that sockpuppets also appear to write worse than ordinary users. They are more likely to swear (0.003 vs 0.002, $p < 0.05$) and use more punctuation (0.057 vs 0.055, $p < 0.05$), while using fewer alphabetic characters (0.769 vs



(a) (b)

Figure 5.6: (a) Histogram for the most active topic for each sockpuppet account. (b) In a sockpuppet group, the secondary sockpuppets tend to be used alongside the primary sockpuppet.

0.771, $p < 0.05$). Sockpuppets also use fewer standard English parts-of-speech, such as articles, verbs, adverbs and conjuctions (*e.g.* for articles, 0.062 vs 0.064, $p < 0.001$). However, while trolls wrote posts that were less readable [78], sockpuppets write posts with similar readability (automated readability index, or ARI = 11.24 vs 11.41 of ordinary users, p=0.09). Sockpuppets also tend to agree more in their posts (0.002 vs -0.012, $p < 0.05$), possibly to minimize conflict with others and support their other sockpuppet account. They also express less negative sentiment (0.022 vs 0.023, $p < 0.001$), though their overal sentiment, subtracting negative from positive sentiment, is similar to that of ordinary users (0.030 vs 0.028, p = 0.43).

5.3.3 Activity and Interactions

Next, we study how sockpuppets interact with the community at large, and how it responds to these sockpuppets.

Sockpuppets start fewer discussions, and post more in existing discussions. First, we note that sockpuppets start fewer discussions, but rather post more within existing discussions (65% of

sockpuppets' posts are replies compared to 51% for ordinary users, $p < 0.001$). This shows that sockpuppets are mainly used to reply to other users.

Sockpuppets tend to participate in discussions with more controversial topics. Do sockpuppets create accounts to participate in certain topics? To answer this, we look at the topics of the discussions in CNN on which sockpuppets post. As shown in Figure 5.6(a) topics that tend to attract more controversy such as *usa, world, politics, justice* and *opinion*, also attract the majority of sockpuppets, while other topics such as health and showbiz have comparatively fewer sockpuppets. This indicates that one of the main motivations for using sockpuppets is to use them to build support for a particular position, corroborating prior work [134, 136].

Sockpuppets are treated harshly by the community. A community can provide feedback to a sockpuppet in three ways: other users can vote on or report their posts, and moderators can delete the posts. Comparing the posts made by sockpuppets with those made by ordinary users, we find that sockpuppets' posts receive a greater fraction of downvotes (0.51 vs 0.43, $p < 0.001$), are reported more often (0.05 vs 0.026, $p < 0.001$) and are also deleted more often (0.11 vs 0.08, $p < 0.001$). Moreover, sockpuppets are also blocked by the moderators more often (0.09 vs 0.07, $p < 0.001$). Overall, this suggests that sockpuppets are making undesirable comments.

Sockpuppets in a pair interact with each other more. Pairs of sockpuppets also tend to post together in more sub-discussions compared to random pairs of ordinary users (6.57 vs 0.33, $p < 0.001$). Moreover, looking at when posts are made, pair of sockpuppets also post more frequently on the same discussion within 15 minutes of each other (7.8 vs 4.28, $p < 0.001$). In other words, pairs of sockpuppets are significantly more likely to interact with one another, and post at the same time, than two ordinary users would.

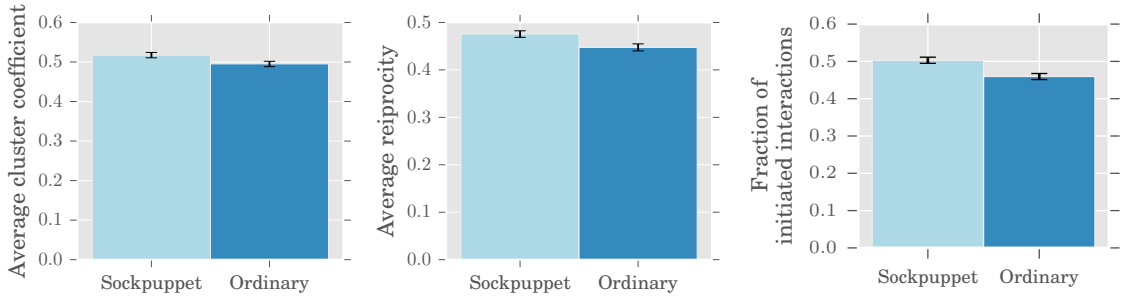
Sockpuppets in a pair upvote each other more. Looking at votes, pairs of sockpuppets vote

significantly more often on each other's posts than random pairs of ordinary users (9.35 vs 0.40 votes, $p < 0.001$). Among the two sockpuppets in a pair, the secondary sockpuppet votes more on primary sockpuppet's posts than vice-versa (14.2 vs 4.5 votes, $p < 0.01$). Moreover, pair of sockpuppets largely give positive votes to each other as compared to ordinary users (0.987 vs 0.952; $p < 0.05$). Altogether, sockpuppets in a pair use their votes to significantly inflate one another's 'popularity'.

Secondary sockpuppets are used in conjunction with primary sockpuppets. Puppetmasters, while controlling multiple sockpuppets, may either use multiple sockpuppets at the same time, or different sockpuppets at different times. To quantify how a puppetmaster may switch between using different sockpuppets, we compute the fraction of consecutive posts made by a particular sockpuppet, and then compute the entropy of this distribution. This way we quantify how much intertwined is the usage of both sockpuppet accounts. For instance, consider two sockpuppets controlled by the same puppetmaster. The puppetmaster first uses S_1 to write 5 posts, then uses S_2 to write 1 post, switches back to S_1 to write 4 more posts, and then finally switches back to S_2 to write 1 more post. The entropy of this post sequence for S_1 is $-\frac{5}{9}\log\frac{5}{9} - \frac{4}{9}\log\frac{4}{9}$, while that for S_2 is $-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}$.

Thus, a lower entropy signifies that a particular sockpuppet is not being used at the same time as the other sockpuppet, while higher entropy indicates that that sockpuppet is being used at the same time as another sockpuppet, with the puppetmaster constantly switching between the two.

Figure 5.6(b) shows the entropy of the primary and the secondary sockpuppets in a sockpuppet pair. We find that secondary accounts tend to have higher entropy, meaning that these sockpuppets are more likely to be used in conjunction with the primary sockpuppet, and thus may be used to support the primary account (e.g., in writing supportive replies). In contrast, primary



(a) (b) (c)

Figure 5.7: Comparison of egonetwork of sockpuppets and similar random users in the reply network.

sockpuppets have lower entropy, meaning they tend to be used more exclusively.

5.3.4 Reply network structure

Last, we examine the user-user interaction network of the entire discussion community. To do this, we create a *reply network*, where a node represents a user and an edge from node A to node B indicates that A replied to B's post at least once. Figure 5.1 shows the reply network of The AV Club discussion community, with red nodes denoting the sockpuppets and blue nodes denoting ordinary accounts. Here, we observe that the nodes denoting sockpuppets are more central in the network. In particular, we find that sockpuppets tend to have higher pagerank than ordinary users (2×10^{-4} vs 1×10^{-6}, $p < 0.001$).

To further understand the differences in how the sockpuppets interact with other users, we additionally compare the ego network of sockpuppets with that of ordinary users (Figure 5.7). We observe that both the number of nodes, and density of the ego networks of sockpuppets and ordinary users are similar (291.5 vs. 291.3 nodes, p = 0.97, and densities of 0.24 vs. 0.22, $p <$ 0.01). However, the ego networks of sockpuppets are more tightly knit, as measured by the average

clustering coefficient (Figure 5.7(a), 0.52 vs 0.49, $p < 0.001$). The nodes in a sockpuppet's ego network reply more to their neighbors, as measured by the average reciprocity (Figure 5.7(b), 0.48 vs 0.45, $p < 0.001$) with sockpuppets generally initiating more interactions (that is, they reply to more users than the users that reply to it, Figure 5.7(c), 0.51 vs 0.46, $p < 0.001$). These observations suggest that sockpuppets are highly active in their local network, and also generate more activity among the other users.

5.4 Types of Sockpuppetry

Different types of sockpuppets exist, and their characteristics suggest that they may serve different purposes. Here, in contrast to prior work which assumes that sockpuppets usually pretend to be other users [34, 36, 37], we find that sockpuppets can differ in their deceptiveness – while many sockpuppets do pretend to be different users, a significant number do not. When sockpuppets participate in the same discussions, they may also differ in their supportiveness – sockpuppets may be used to support other sockpuppets of the same puppetmaster, while others may choose not to.

5.4.1 Deceptiveness: Pretenders vs. non-pretenders

A pair of sockpuppets can pretend to be two separate individuals, or may simply be two user accounts an individual uses in different contexts, without any masquerading. We refer to the former group of sockpuppets as *pretenders*, and the latter group as *non-pretenders*.

One way we might quantify the deceptiveness of a sockpuppet pair is to examine the similarity of display names and email addresses (we only examine the part of the email address before the @-sign). Display names are public and show up next to user's comments, while email addresses are private and only visible to forum administrators. If a pair of sockpuppets wants to appear as two separate users, each may adopt a display name that is substantially different from

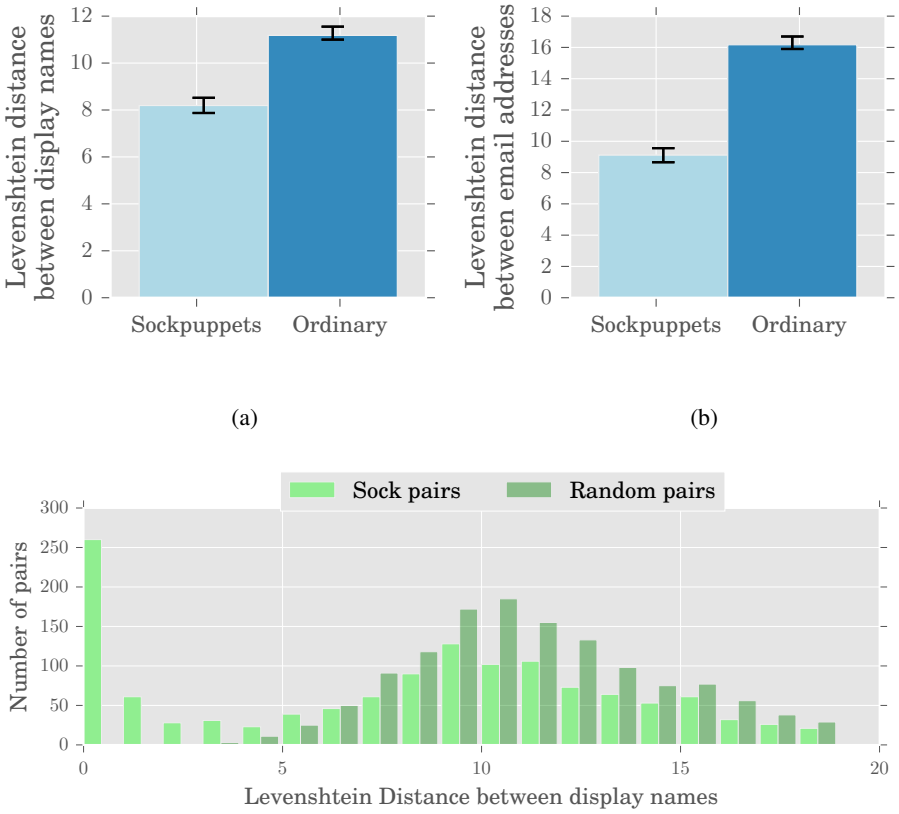


Figure 5.8: The (a) display names and (b) email addresses of the sockpuppet accounts are more similar to each other compared to similar random pairs. (c) Based on the distance of display names, sockpuppets can be *pretenders* (high distance) or *non-pretenders* (low distance).

the other in order to deceive community members. Puppetmaster may also adopt significantly different email addresses to avoid detection by system administrators. To quantify this difference, we measure the Levenshtein distance between two display names, as well as the corresponding email addresses.

Figures 5.8(a) and 5.8(b) compare how display names and email addresses differ between pairs of sockpuppets, and between random pairs of ordinary users. We observe that sockpuppets pairs have both more similar display names as well as email addresses than what would be expected by comparing random pairs of users. This also serves as evidence that sockpuppets we

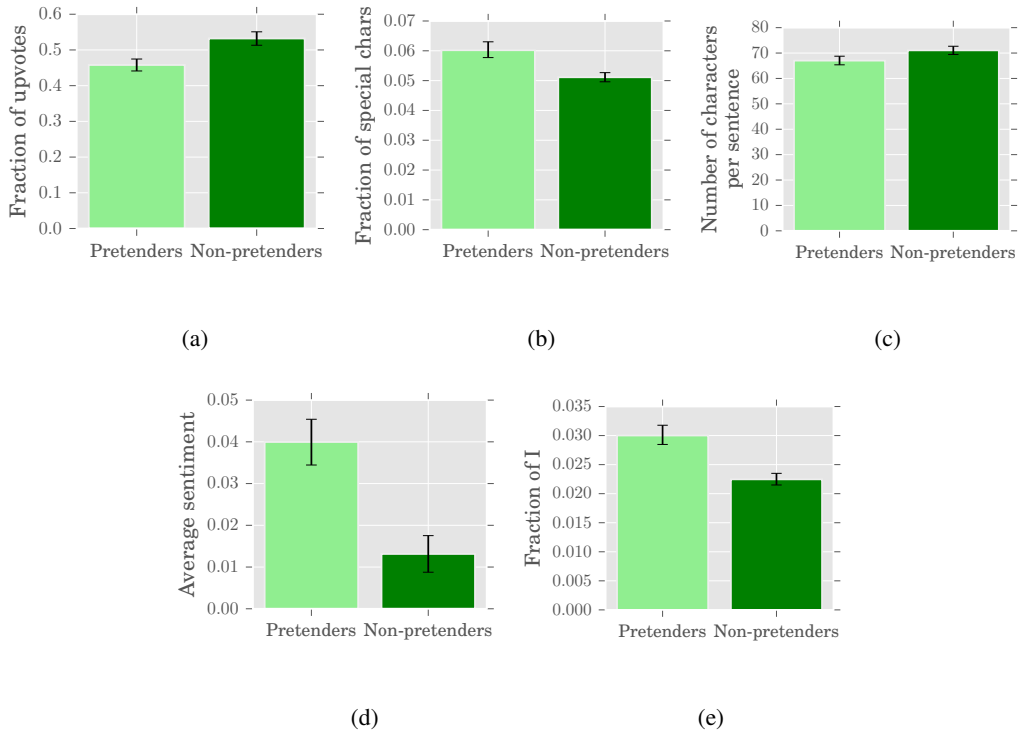
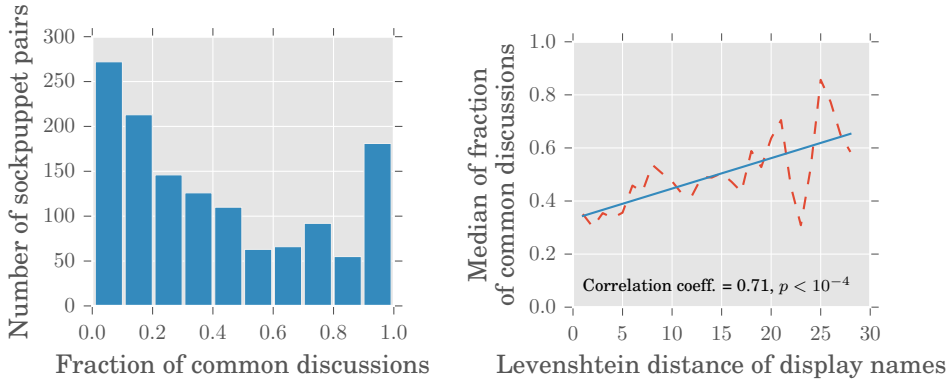


Figure 5.9: Differences between pretenders and non-pretenders: (a) fraction of upvotes, (b) fraction of special characters in posts, (c) number of characters per sentence, (d) average sentiment, (e) usage of first person pronoun ("I").

identified are likely to have been created by the same individual. Further, we also observe that email addresses of sockpuppets are 50% more similar than those of ordinary accounts, while display names of sockpuppets are only 25% more similar than expected at random. This observation may be explained by the fact that sockpuppets put more effort into picking unique display names, which are public-facing, and less effort into picking unique email addresses, which are private-facing and less likely to be noticed.

But are all sockpuppets simply more likely to have more similar display names? Examining the distribution of the distances between display names in Figure 5.8(c), we find that the distribution for random pairs is unimodal, while for sockpuppets it is bimodal. This bimodality suggests that two types of sockpuppets pairs exist. The first type of sockpuppets has virtually identical



(a) (b)

Figure 5.10: (a) Based on the fraction of common discussions between sockpuppet pairs, there are two types of sockpuppets: *independent*, which rarely post on same discussion, and *sock-only*, which only post on same discussions. (b) Increase is display name distance is highly correlated with discussion use.

display names (Levenshtein distance < 5), and these are what we call non-pretenders. The second type of sockpuppets has substantially different display names (Levenshtein distance ≥ 5), and we call them pretenders. Pretenders are likely to be created for deception and use different display names to avoid detection. Non-pretenders on the other hand have similar display names and this may implicitly signal to the community that they are controlled by the same individual, and thus may be less likely to be malicious.

Across all communities, we find 947 pretender and 403 non-pretender sockpuppet groups. We observe that pretenders tend to participate in the same discussions. For example, Figure 5.10(a) plots the fraction of common discussions over all sockpuppet groups. We observe bimodality here as well, which may be partially explained by the bimodality of the distribution of display name distances – Figure 5.10(b) shows that the likelihood of a pair of sockpuppets participating in the same discussion increases as their display names become more different (fitted regression line shown in solid for clarity). In other words, these observations suggest that sockpuppets that

participate in many common discussions have very different display names (high Levenshtein distance) and are thus pretenders, while accounts that participate in few common discussions tend to have similar display names and are thus non-pretenders.

Figure 5.9 additionally illustrates the other differences of pretenders and non-pretenders. We find that pretenders' posts are both more likely to be reported (0.06 fraction of all pretenders' posts are reported vs 0.03 for non-pretenders, $p < 0.001$), be deleted by moderators (0.11 vs 0.08, $p < 0.001$), and receive a smaller fraction of up-votes (0.45 vs 0.53, $p < 0.001$). Pretenders also write posts that contain more uppercase (0.07 vs 0.05, $p < 0.001$) and special characters (0.06 vs 0.05, $p < 0.001$), which suggests both shouting, as well as swearing. In contrast, non-pretenders wrote posts which were longer (35.2 words vs 38.6, $p < 0.05$) and more readable (ARI 11.15 vs 11.58, $p < 0.05$). Pretenders' posts also contained more positive sentiment (0.04 vs 0.013, $p < 0.001$) and agreement words (0.006 vs 0.005, $p < 0.001$), suggesting that they tended to be more affable.

5.4.2 Supporters vs. Dissenters

Prior work suggests that a primary purpose of sockpuppets is to sway public opinion by creating consensus [8]. Thus, we focus our attention on sockpuppets participating in the same discussion, and examine how they interact with each other – do sockpuppets tend to support each other?

We study two ways in which a sockpuppet pair (S_1, S_2) may interact – directly, where one sockpuppet (S_2) replies to another sockpuppet (S_1), or indirectly, where one sockpuppet (S_2) replies to a third user (O) who had replied to the first sockpuppet (S_1).

We focus on the extent to which sockpuppet S_2 agrees with sockpuppet S_1, and measure agreement as the difference between fraction of words categorized by LIWC as assenting and those

categorized as either negations or dissenting [143]. We additionally adjust the sign of agreement depending on who the replying sockpuppet is replying to. For example, S_2 may write a post disagreeing with an ordinary user O. But if that ordinary user O in turn disagreed with the initial sockpuppet S_1, then we assume that the replying sockpuppet S_2 is instead in agreement with the initial sockpuppet S_1. We divide sockpuppets into three groups – *supporters*, who have a positive agreement score, *non-supporters*, who have an agreement score of zero, and *dissenters*, who have a negative agreement score.

Across all communities, we find that 60% of the sockpuppets are non-supporters, while 30% are supporters. Only 10% of sockpuppets are dissenters. Examining these discussions, we find evidence that supporters tend to support the arguments of S_1 (e.g., 'I agree, or 'so true'), and sometimes make additional arguments in their favor (e.g., 'That will cost him the election [...]').

On the other hand, dissenters tend to argue against S_1 (e.g., 'That's not what you said [...]'). We hypothesize that one reason sockpuppets may disagree with each other, despite being controlled by the same puppetmaster, may simply be to attract more attention to the argument. In some cases, we observed a dissenter making easily refutable arguments (e.g., 'Ok if your [sic] so worried about being spied Throw away all your electronics.'), which may have served to discredit the opposing view.

Altogether, these observations suggest that within discussions, sockpuppets may adopt different roles. While most sockpuppets argue for other sockpuppets controlled by the same puppetmaster, a small but significant number instead argue against other sockpuppets instead.

Nonetheless, is there a relationship between deceptiveness and supportiveness? Figure 5.2 shows that overall, users who support other users in discussions are most likely to be also pretending to be other users (74% of supporters are pretenders). Interestingly, when users dissent in a discussion, they are less likely to be a pretender. This suggests that pretending is most important

	Pretender	Non-pretender
Supporter	0.74	0.26
Non-supporter	0.70	0.30
Dissenter	0.58	0.42

Table 5.2: 74% of supporters, 70% of non-supporters and 58% of dissenters are pretenders.

when a puppetmaster is trying to create an illusion of consensus.

5.5 Experimental Evaluation: Detecting Sockpuppets

Our previous analysis found that sockpuppets generally contribute worse content and engage in deceptive behavior. Thus, it would be useful to create automated tools that can help identify sockpuppets, and assist moderators in policing online communities. In this section, we consider two classification tasks, both of which relate to the prediction of sockpuppetry. First, can we distinguish sockpuppets from ordinary users? And second, can we identify pairs of sockpuppets in the communities?

Based on the observations and findings from the analyses in the previous sections, we identify three sets of features that may help in finding sockpuppets and sockpuppet pairs: activity features, community features, and post features. For each user U, we develop the following features:

Activity features: This set of features is derived from U's posting activity. Prior research has shown that activity behavior of bots, spammers, and vandals is different from that of benign users [13, 30, 37, 144, 145]. Moreover, in our analysis, we have seen that sockpuppets make more posts and they start less sub-discussions. Therefore, the activity features we consider include the number of posts, the proportion of posts that are replies, the mean time between two consecutive

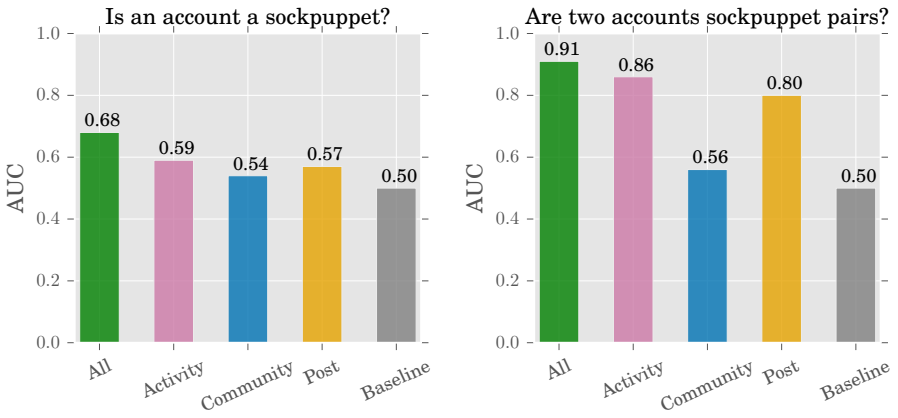
Feature Set	Features
Activity	Reply egonetwork clustering coefficient and reciprocity,
	Number of posts, proportion of reply posts,
	Time between posts, tenure time
Community	Whether account is blocked, fraction of upvotes,
	Fraction of reported and deleted posts
Post	Number of characters, syllables, words, sentences,
	Fraction of punctuations, uppercase characters, etc.,
	Number of syllables per word, words per sentences, etc.
	Readability metrics (*e.g.* ARI), LIWC (*e.g.* swear words),
	Agreement, sentiment and emotion strength

Table 5.3: Three sets of features were used to identify sockpuppets and sockpuppet pairs.

posts, and U's site tenure, or the number of days from U's first post. Further, we use features based on how U is situated in the reply network. Here, U's local network consists of U, the users whose posts U replied to, and the users that replied to U's posts. We then consider clustering coefficient and reciprocity of this network. In addition, for the task of identifying pairs of sockpuppets, we use number of common sub-discussions between these sockpuppets to measure how often the two comment together.

Community features: Interactions between a user and the rest of the community may also be indicative of sockpuppetry. Community feedback on an account's posts has been effective in identifying trolls and cheaters [78, 146], and we also observed that sockpuppets are treated more harshly than ordinary users. Thus, we consider the fraction of downvotes on posts U wrote, as well as the fraction that were reported or deleted, in addition to whether U was blocked.

Post features: Finally, we also measure the linguistic features of U's posts. These features have been very effective to identify sockpuppets [34, 134], authors of text [138, 147], deceptive



(a) (b)

Figure 5.11: Classification performance to identify (a) sockpuppets from ordinary users and (b) pairs of sockpuppet accounts (bottom). Activity features have the highest performance.

writing styles [32], trolls [78], hoaxes [30], and vandalism [142]. In our analysis, we observe that sockpuppets do indeed write differently, for example, writing shorter sentences, using more swear words, and using more singular first-person pronouns. Thus, we incorporate linguistic features such as the number of characters, average word length, average number of syllables in words, number of big sentences, text readability (measured using ARI), and the different categories of LIWC features. Finally, we also consider sentiment, emotional valence, and agreement (as described previously).

5.5.1 Is an account a sockpuppet?

Given the posts a user has made, is it possible to distinguish a sockpuppet from an ordinary user? To control for user activity, we match sockpuppets and ordinary users on their total number of posts, as well as the discussions they post in. Matching gives a balanced dataset, where random guessing results in 50% accuracy. We perform 10-fold cross validation using a random forest classifier, then measure performance using ROC AUC.

Figure 5.11(a) shows results obtained individually with each feature set, as well as when considering all the features together. We observe that when using all features, the AUC is 0.68. Individually, all three feature sets perform similarly, with activity features slightly more predictive than the others (AUC=0.59).

To find the relative advantage of adding features, we perform forward feature selection. We observe that activity and post features perform close to the final AUC of 0.68, and that there is not much lift by adding the community features. This means that to identify sockpuppets, their activity and content of the post matter the most.

5.5.2 Are two accounts sockpuppet pairs?

Next, we turn our attention to identifying pairs of sockpuppets. Given a sockpuppet and two other users, where one is a sockpuppet, can we predict which user is the sockpuppet?

For each sockpuppet pair (S_1, S_2), we choose a matching ordinary account O for sockpuppet S_1. Again, this results in a balanced dataset and the task is to identify which of these two pairs is a sockpuppet. Features used in this experiment are the differences in the individual feature values for the two accounts each pair. We again evaluate performance using a random forest classifier.

Figure 5.11(b) shows the performance of the resulting classifier. We achieve a very high AUC of 0.91, suggesting that the interactions between sockpuppets are strongly predictive. Looking at the individual features, we see that activity features again perform the best, with close to 0.86 AUC. Community features perform the poorest, with an AUC of 0.56.

Overall, our results suggest that activity-based features best predict sockpuppetry. While it is possible to differentiate sockpuppets from ordinary users, it is significantly easier to find other sockpuppets in the same group once at one sockpuppet has been identified. The latter result

suggests most importantly, the interactions between sockpuppets are the best way to identify them.

5.6 Discussion and Conclusion

Our findings shed light on how sockpuppets are used in practice in online discussion communities. By developing a robust methodology for identifying sockpuppets, we are able to comprehensively study their activity. Importantly, this methodology is able to identify sockpuppets that were created at significantly different times, use very different usernames or email addresses, write differently, or mostly post in different discussions.

Our work revealed differences in how sockpuppets write and behave in online communities. Sockpuppets use more singular first-person pronouns, write shorter sentences, and swear more. They participate in discussions with more controversial topics, and are especially likely to interact with other sockpuppets. These differences allowed us to build preditive models that robustly differentiate pairs of sockpuppets from ordinary users, as well as identify individual user accounts that are sockpuppets.

By developing better techniques to identify sockpuppets, our work can be used to improve the quality of discussions online, where each discussant can better trust that their interactions with others are genuine. Nonetheless, while it is possible to identify sockpuppetry, one should be careful not to assume that all sockpuppets are malicious. Our work suggests that a significant number of sockpuppets do not pretend to be other users and were simply used to participate in different discussions. This observation suggests that some users find it valuable to separate their activity in different spheres of interests.

Part III

Graph Based Algorithms for Malicious Behavior Detection

Chapter 6

Detecting Trolls in Slashdot

In this chapter, we analyse trolls in the Slashdot social network. The uniqueness of this social network is that it allows users to mark each other as 'friend' or 'foe', thereby creating an explicit signed network, where positive links represent the friend relation and negative links represent the foe relation. Given that a user can create connections to other users, it is natural that malicious users create links to promote and hide themselves. We will explore a decluttering operation applicable to graphs, that removes irrelevant edges from the network. Since not all edges are important in the network, this operation works by identifying and keeping only the important ones. This is the first of the two graph-based algorithms that we will look at in this thesis.

6.1 Introduction

A signed social network (SSN) is one in which a user u can have a positive or negative relationship with another user v. There are many signed social networks in the real world. Even in small human populations (e.g. faculty in a computer science department), there will be individuals who like some individuals but dislike others. In some online networks like Slashdot, users may explicitly mark some users as friends and others as foes. On Wikipedia, an individual may

"roll back" or "reverse" essential changes made by one person, while supporting and augmenting changes by another. An implicit negative opinion is conveyed in the first case and a positive opinion in the latter case. On Twitter, a user u may frequently support what a user v_1 says while opposing or contradicting what another user v_2 says.

In this chapter, we start with a "Signed Social Network (SSN)" $G = (V, E, W)$ where V is a set of users, $E \subseteq V \times V$ is a set of edges, and $W : E \rightarrow [-1, +1]$ assigns a real valued weight from -1 to +1 indicating how positive or negative one user is to another. When $W(u, v) = 1$, u considers v to be a 100% friend, when $W(u, v) = -1$, he considers v to be a 100% foe. While SSNs are explicitly present in Wikipedia and Slashdot Zoo, they can also be extracted via NLP techniques from Twitter. *We restrict this chapter to network analysis and assume a signed network is given as input. Clearly, methods to extract signed networks from networks like Twitter is an important task - but is not addressed here.*

A *malicious user* in an SSN $G = (V, E, W)$ is a specially designated individual. On Slashdot, trolls are malicious users who post or spread misleading, offensive or nonsensical information on the network. Likewise Wikipedia describes a vandal as "an editor who intentionally makes unconstructive edits to Wikipedia's pages." Vandals may insert irrelevant information, nonsense, obscenity or crude humor to pages or entirely blank or delete pages. A *benign user* is one who is not malicious.

The goal of this chapter is to present a single framework within which to identify malicious users. A major challenge in effectively identifying trolls is the fact that malicious users take a number of carefully designed steps that enable them to evade detection. We propose *5 graph decluttering operations* that help simplify a large, complicated SSN G into a smaller and simpler SSN G'. Intuitively, the idea is to remove some "hay" from the "haystack" we are searching in order to present our TIA algorithm with a simpler signed graph, stripped of irrelevant edges, that

enables TIA to operate more effectively. We tested all subsets of these 5 decluttering operations and found the combination that yields the best results.

Most CS work on signed networks have involved study of Slashdot, Epinions and Wikipedia administrator election networks. We focus only on Slashdot, as there is no ground truth present for malicious vs benign users on Epinions, and because NLP is needed to analyze Wikipedia.

The chapter is organized as follows. Section 6.2 briefly defines Signed Social Networks (SSNs). Section 6.3 briefly looks at how centrality measures on SSNs have been used in the past in order to identify trolls on Slashdot Zoo and explains both these centrality measures as well as other centrality measures. Section 6.4 presents our 5 decluttering operations to simplify a complex SSN into a smaller and simpler SSN - Section 6.5 presents the TIA algorithm that uses a subset of these decluttering operations to simplify an SSN. We present the Slashdot Zoo data set we used in Section 6.6.1. Section 6.6 reports on the results of experiments to assess how well TIA works. We examine how the combination of decluttering and a Signed Eigenvector Centrality (SEC) measure together generate the best accuracy results on the Slashdot data set. We show that under appropriate settings,TIA has: (i) over 3 times the precision of the best existing algorithm to find trolls in Slashdot [148], (ii) retrieves over twice the number of trolls that [148] does, and (iii) does all this while running 25-50 times faster.

6.2 Signed Social Networks

A *Signed Social Network (SSN)* is a directed, weighted graph $G = (V, E, W)$ where V is a set of users, $E \subseteq V \times V$ is a set of edges, and $W : E \rightarrow [-1, +1]$ is a mapping. $W(u, v)$ can be thought of as assigning a "likes" or a "friendship" score describing how much user u likes a user v.

Slashdot: On Slashdot, a user u can explicitly mark a user v as a foe (i.e. $W(u, v) = -1$) or as a

friend (i.e. $W(u,v) = +1$).

Wikipedia: On Wikipedia, we can define $W(u,v)$ using NLP in many ways. One way is to set

$$W_{wi}(u,v) = \frac{supp(u,v) - rev(u,v)}{edits(v)}$$

where $edits(v)$ is the set of all edits made by v during some fixed time window, $supp(u,v)$ is the number of edited documents in $edits(v)$ that were subsequently edited but not substantively reverted by v, and $rev(u,v)$ is the number of edits in $edits(v)$ that were subsequently reversed by u. A more sophisticated way to define $W(u,v)$ is given in [149].

YouTube: One way to derive an SSN from YouTube is to set:

$$W_{yt}(u,v) = \frac{thumbs\_up(u,v) - thumbs\_down(u,v)}{|posts(v)|}$$

where $posts(v)$ is the set of all videos posted (during some fixed time frame) by v that were marked positively or negatively by u, $thumbs\_up(u,v)$ is the number of videos in $posts(v)$ marked with a "thumbs up" and "thumbs_down" is the number of videos in $posts(v)$ marked with a thumbs down.

Twitter/Facebook: Given an edge (u,v) denoting that u follows v on Twitter (or friends v on FB), we set:

$$W_{tw}(u,v) = \frac{pos(u,v) - neg(u,v)}{|tweets(v)|}$$

where $tweets(v)$ is the set of tweets posted by v during a given time frame, $pos(u,v)$ is the subset of those tweets that are either retweeted by u or essentially rephrased by u afterwards, and $neg(u,v)$ is the subset of $tweets(v)$ that are the sets of tweets in $pos(u,v)$ whose content is contradicted by a subsequent post by v. Note that NLP techniques that use sentiment analysis [150, 151] can be used to estimate the sets $pos(u,v)$ and $neg(u,v)$.

Stack Overflow: Users of Stack Overflow can mark comments provided by other users as good ($+1$) or not good (-1).

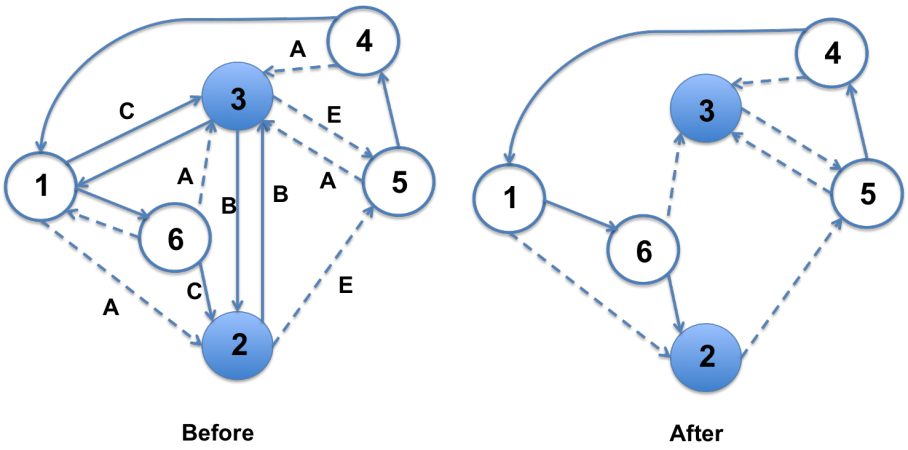


Figure 6.1: (Left) Example of signed social network. Filled nodes are trolls, non-filled nodes are benign users. Solid (resp. dashed) edges mean positive (resp. negative) endorsements. Edges labels are the attack models used by trolls. (Right) resulting SSN after decluttering operations (a) and (d) using SEC.

A SSN can be represented as an adjacency matrix A with values $A_{uv} = W(u, v) \in [-1, +1]$. We use H to denote the transition matrix obtained from A by dividing each non-zero element A_{uv} by $\sum_w |A_{uw}|$.

Example 1 *Figure 6.1 (left) shows a small toy social network which has two trolls — nodes 2 and 3 in it.* □

6.3 Related Work

Signed networks have been studied both in social science ([152]) and computer science ([153, 154]). Much work on reputation systems propagates trustworthiness of users on social networks – but this work has been done on unsigned social networks ([91, 92]). [93] studied the propagation of trust and distrust in social network for the purpose of edge sign prediction. In this section, we will look at the existing centrality measures, the axioms for good centrality measures and attack models.

6.3.1 Centrality Measures for SSNs

Given a SSN $G = (V, E, W)$, a node centrality measure is a function $R : V \rightarrow R$ that assigns a score to each user — the higher the score, the more important the user. Various authors have used different centrality measures to identify malicious users (e.g. [148] uses Negative Rank to identify trolls in Slashdot). We now review some of these measures.

Freaks The freak score of u is the number of incoming negative edges [148]. For weighted networks, the *Freak Centrality* of u is the sum of the weights of incoming negative edges.

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

According to this definition, node 3 in Figure 6.1 has freak centrality 3 because it has 3 incoming negative edges, while nodes 4 and 6 have freak centrality 0 because they have no incoming negative edges.

Fans Minus Freaks (FMF) The centrality of u is the number of positive incoming edges (*fans*) minus the number of negative incoming edges (*freaks*) [148]. For weighted networks, we define *FMF* centrality as the total positive incoming weight minus the the total negative incoming weight.

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

Node 3 in Figure 6.1 has FMF centrality -1 because it has 3 incoming negative edges and 2 incoming positive edge. Node 5 has an FMF centrality of -2. A similar measure, called *Prestige*, has been proposed in [155] and is obtained by dividing the FMF centrality by the sum of the absolute values of the incoming weights for each node.

PageRank (PR) PageRank ([156]) is defined for directed graphs with non-negative edge weights. It was originally developed for indexing web pages, and represents the likelihood that a person following links will arrive at a particular page. The PageRank of a node u is defined as

$$PR(u) = \frac{1-\delta}{|V|} + \delta \sum_{v \in pred(u)} \frac{PR(v)}{|succ(v)|}$$

Here, δ is a "damping factor" (usually 0.85) which captures the probability that a user arrives at a web page by following links (as opposed to landing on the page via some other process). $pred(u)$ is the set of all vertices v such that $(v, u) \in E$ and $succ(v)$ is the set of all vertices v' such that $(v, v') \in E$. Node 3 in Figure 6.1 has a PageRank of 0.29, while node 5 has a PageRank of 0.18. A *Modified PageRank* (M-PR) has been proposed in [157] to take into account both positive and negative links. In particular, they apply PageRank separately on A^+ (sub-network with positive links) obtaining PR^+, and on A^- (sub-network with negative links) obtaining PR^-. The final rank vector M-PR is computed as M-PR $= PR^+ - PR^-$. Nodes 3 and 5 in Figure 6.1 have M-PR scores of -0.09 and -0.41, respectively.

Signed Spectral Ranking (SSR) Signed Spectral Ranking (SSR) [148] improves upon PageRank by taking edge signs into account. It is computed by taking the dominant left eigenvector of the signed matrix

$$G_S = \delta \cdot H_A + \frac{(1-\delta)}{|V|} \cdot J_{|V| \times |V|}$$

Positive edges correspond to endorsements, while negative edges to criticisms. Node 3 in Figure 6.1 has an SSR of 0.74, while node 5 has an SSR of -0.50.

Negative Ranking (NR) An empirical evaluation of SSR and PR using Slashdot data was done in [148] who show that SSR and PR values were almost equivalent for benign users, but PR value for trolls was much more than their SSR value. They suggest a Negative Rank measure computed by subtracting PR from SSR, i.e. $NR(u) = SSR(u) - \beta \cdot PR(u)$, where β is a parameter determining the influence of PageRank on the ranking. As [148] obtained their best results when $\beta = 1$, we use $\beta = 1$. Node 3 in Figure 6.1 has a NR of 0.45, while node 5 has a NR of -0.68.

Signed Eigenvector Centrality (SEC) Eigenvector centrality (EC) was proposed by Bonacich [158] for networks with non-negative edge weights given by the dominant eigenvector of the adjacency matrix. As eigenvectors can be computed for any matrix, [152] suggests that this measure can also be computed for (weighted) signed networks. Thus, the signed eigenvector centrality of a vertex v can be computed from the vector x that satisfies the equation $Ax = \lambda x$, where λ is the greatest eigenvalue. According to this definition, node 3 in Figure 6.1 has SEC of 0.68, while node 5 has an SEC of -0.55.

Modified HITS (M-HITS) The HITS link analysis algorithm to rate Web pages [159] has been adapted for SSNs in [157] by iteratively computing the hub and authority scores separately on A^+ and A^-, using the equations:

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

and by assigning, after convergence, the score $a(u) = a^+(u) - a^-(u)$ to each node u. $pred^+(u)$ (resp. $pred^-(u)$) denotes the set of nodes v in $pred(u)$ s.t. $W(v,u) > 0$ (resp. $W(v,u) < 0$). Similarly for $succ^+(u)$ and $succ^-(u)$. For M-HITS, node 3 in Figure 1 has score of -0.92 and node 5 has score -9$\times 10^{-9}$.

Bias and Deserve (BAD) In [52], a node u's bias (BIAS) reflects the expected weight of an outgoing connection, while its deserve (DES) reflects the expected weight of an incoming connection from an unbiased node. Similarly to HITS, BIAS and DES are iteratively computed as:

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

where $X^t(v,u) = \max(0, BIAS^t(v)W(v,u))$. Finally, the scores of the nodes in the SSN are taken as the vector DES. In Figure 1, nodes 3 and 5 have BAD scores -0.16 and -1.0 respectively.

Example 2 *Consider the SSN in Figure 6.1 (left) — nodes* 2 *and* 3 *are trolls. The following table (left part) shows how nodes are ranked according to the centrality measures. Here, the row "Freaks" should be read as: In the original network, the Freak centrality of node 3 is lowest, followed by 5, followed by 1 and 2 (with same freaks centrality) and 4,6 (with same freaks centrality). The rest of this row can be read similarly for the decluttered network (discussed later).*

Measure	*Original network*						*Decluttering with* $\{a,b,d\}$					
	Lowest			*Highest*			*Lowest*			*Highest*		
Freaks	***3***	*5*	*1,2*		*4,6*		***3***	*5*	*1,2*		*4,6*	
FMF	*5*	***3***	*1,2,4,6*				*5*	***3***	*2,4,6*			*1*
Prestige	*5*	***3***	*1,2*		*4,6*		*5*	***3***	*2*	*1,4,6*		
M-PR	*5*	***3***	*4*	*6*	*1*	*2*	*5*	***3***	*4*	*6*	*1*	*2*
SSR	*5*	*4*	*6*	*1*	*2*	*3*	***3***	***2***	*6*	*1*	*4*	*5*
NR	*5*	*4*	*1*	*6*	*2*	*3*	***3***	***2***	*6*	*1*	*4*	*5*
SEC	*5*	*4*	*6*	*1*	*2*	*3*	***3***	***2***	*6*	*1*	*4*	*5*
M-HITS	***3***	*5*	*4*	*6*	*1*	*2*	***3***	*5*	*4*	*6*	*1,2*	
BAD	*5*	***3***	*2*	*1*	*4,6*		*5*	***3***	*2*	*1*	*4,6*	

If we take the two lowest (as there are two trolls) scored nodes to be trolls, then Freaks, FMF, M-PR, Prestige and BAD identify one troll (node 3) correctly and incorrectly identify node 5, among the lowest two nodes. □

It is easy to construct cases where Freak Centrality cannot identify a node as a troll. For instance, consider a network with 1000 nodes including a node A which has 995 positive incoming edges and 4 negative edges. All other nodes have 5 incoming positive edges and either 0 or 1 incoming negative edges. Clearly, A would be designated as having the highest freaks centrality — but it is not likely to be a troll because the 995 positive incoming edges far outweigh the 4 negative incoming edges.

6.3.2 Requirements of a good scoring measure

[152] proposes a set of axioms for SSNs that a good measure of centrality in SSNs must satisfy under the assumption that a set of nodes is benign (and the others are malicious).[1]

Axiom 1: *A positive edge from a benign node to a node v should increase v's centrality.* Intuitively, a positive edge from a benign node to another node means that the benign node also thinks the other node is benign — otherwise there is no reason for the benign node to implicitly endorse the other node.

Axiom 2: *A negative edge from a benign node to a node v should decrease v's centrality.* As in the previous axiom, benign nodes have an incentive to identify malicious nodes in order to preserve the integrity of the social network as a whole. As a consequence, when a benign node says a node is malicious, there is some chance that it actually is malicious.

Axiom 3: *A positive edge from a malicious node to a node v should decrease v's centrality.* The rationale behind this axiom is that malicious nodes have a strong incentive to endorse other malicious nodes so that an "army" of malicious nodes can collectively perform some task(s).

Axiom 4: *A negative edge from a malicious node to a node v should increase v's centrality.* As in the previous case, malicious nodes have an incentive to downgrade the centrality of benign nodes so that, in comparison, other malicious nodes get a high score. As a consequence, when a node is disliked by a malicious node, it probably means that the malicious node is trying to "bad mouth" a benign node.

Table 6.1 shows which centrality measures satisfy these axioms (in one iteration of computing the measure). A *Cond-Yes* means that under some reasonable conditions, the centrality measure satisfies the axiom in question. For instance, Freaks and FMF centrality measures do not

[1] Our TIA algorithm will make such assignments initially and then iteratively modify these assignments in each iteration of a loop.

distinguish between the origin of an edge and hence they do not satisfy some axioms — in both cases, each negative incoming edge decreases the centrality of a vertex irrespective of the origin of the edge (same thing happens with Prestige, M-PR, M-HITS and BAD). PageRank does not take the sign of an edge into account and so a negative incoming edge may increase a user's centrality. Signed Spectral Rank and Signed Eigenvector Centrality both satisfy all the four axioms as long as they assign positive and negative centrality scores to benign and malicious users, respectively. If this condition is not satisfied, then these measures violate the axioms. These two centrality measures take into consideration the centrality value of the edge generator and the sign of the edge. As Negative Rank depends on the relative increase in the values of PR and SSR, there can be some indecisive cases. Consider Axiom 1 – if the conditions stated above hold, then both Page Rank and Signed Spectral Rank would increase. The decision for Negative Rank depends on the relative increase in the centrality values and therefore, nothing can be said in general. Similarly for Axiom 4.

6.3.3 Attack Models

In the real world, benign users can be tricked into endorsing malicious users (via positive outgoing edges). For example, benign users may endorse someone because they were endorsed by that user, not necessarily because they like that user. Malicious users may endorse a benign user (i.e. have positive edges to benign users) in the hope that the benign user reciprocates, which would increase their centrality. In such cases, past methods to identify malicious nodes in SSNs can lead to error as shown in Example 2 and the following discussion. Specific attacks described in [77] include:

(A) Individual malicious peers. Malicious users always present bad behavior, and hence receive negative links from good users. These are relatively stupid malicious users who should not be

Ranking	*Axiom 1*	*Axiom 2*	*Axiom 3*	*Axiom 4*
Freaks	No	Yes	No	No
FMF	Yes	Yes	No	No
Prestige	Yes	Yes	No	No
PR	Yes	No	No	Yes
M-PR	Yes	Yes	No	No
SSR	Cond-Yes	Cond-Yes	Cond-Yes	Cond-Yes
NR	Can't Say	Cond-Yes	Cond-Yes	Can't Say
SEC	Cond-Yes	Cond-Yes	Cond-Yes	Cond-Yes
M-HITS	Yes	Yes	No	No
BAD	Yes	Yes	No	No

Table 6.1: Table showing which axioms are satisfied by the centrality measures. Yes, No and Cond-yes mean that the axioms are satisfied, not satisfied and conditionally satisfied, respectively. Can't say means that nothing can be said in particular.

difficult to detect, say by using Freaks centrality.

(B) Malicious collectives. Malicious users endorse other malicious users. In this case, a malicious user's score may increase due to the presence of a bunch of positive incoming links.

(C) Camouflage behind good transactions. Malicious users can cheat some benign users to vote positively for them. This happens, for instance, when malicious users endorse a benign user who, out of courtesy, endorses them back.

(D) Malicious spies. There are two kinds of malicious users: some of them act as in threat models B and C, while the others (called *spies*) make benign users to vote positively for them, and assign positive value only to bad nodes.

(E) Camouflage behind judgements. The strategy in this case is to assign negative value to good users. Then, this can cause the decrease of rank for good peers, and, consequently, the increase of malicious user's rank.

Measures	*A*	*B*	*C*	*D*	*E*
Freaks	Yes	No	No	No	No
FMF	Yes	No	No	No	No
Prestige	Yes	No	No	No	No
PR	No	No	No	No	Yes
M-PR	Yes	No	No	No	No
SSR	Cond-Yes	Cond-Yes	Cond-No	Cond-No	Cond-Yes
NR	Cond-Yes	Cond-Yes	Cond-No	Cond-No	Can't say
SEC	Cond-Yes	Cond-Yes	Cond-No	Cond-No	Cond-Yes
M-HITS	Yes	No	No	No	No
BAD	Yes	No	No	No	No

Table 6.2: Table showing which centrality measure successfully prevents malicious users from using the attack models *A*-*E*. Yes and Cond-Yes means the attack is always and conditionally prevented, respectively, while No and Cond-No mean the opposite. Can't say means nothing can be said in general.

A good scoring measure should robustly counter all these 5 attack models. Countering an attack model means preventing increase in a malicious user's centrality and decrease in a benign user's centrality. This way malicious nodes following these attack models would not be able to "game the system" and their scores would still be low. Figure 6.1 (left) shows how trolls in our toy example use the attack models (edge labels show the attack model used).

Table 6.2 depicts which centrality measures are resilient to which attacks. A Yes means the measure is able to counter the attack model. Since Freaks and FMF do not take the score of the origin of an edge into account, they both only disable attack model A (so is the case with Prestige, M-PR, M-HITS and BAD). PageRank ignores the sign of the edge, so it only counters attack model E as the centrality of a benign user would increase even when there are incoming negative edges. Signed Spectral Rank and Signed Eigenvector Centrality conditionally deflect ("Cond-Yes") attack models A, B, and E. The condition is that the centrality measure should assign a positive centrality

score to benign users and a negative centrality score to malicious users. If the condition is satisfied, then the attack model is countered, otherwise it could be successful. For instance, if the centrality measure assigns a negative score to a malicious user, then malicious users using attack model B would fail as it would increase the centrality of a positive edge recipient. "Cond-No" means a conditional No. If the above condition is satisfied, then the attack model is successful, otherwise the attack model may fail. For instance, if the condition is satisfied and the centrality measure assigns a positive score to a benign user, then the centrality of malicious users following attack model C would increase and the attack would succeed. Negative Rank deflects an the attack model only if the edge increases the PageRank of the user and decreases her Signed Spectral Rank. If both increase or decrease, then nothing can be said in general.

As no centrality measure successfully handles all the attack models described above, there is a critical need for a mechanism to prevent malicious users from increasing either their centrality or another malicious user's centrality and prevent the decrease of benign users' centrality.

6.4 Decluttering Operations

In order to handle the five attack models described above, we present 5 graph decluttering operations that help reduce the impact of such attacks. Our TIA algorithm (presented after the five operations) iteratively uses the score provided by a centrality measure to identify both benign and malicious users, and removes some edges between benign users, so that, at the end of the process, user scores enable us to better recognize malicious users. Given a centrality measure $\mathcal{C}$ and a threshold value τ, benign users are those nodes v in V s.t. $\mathcal{C}(v) \geq \tau$ — everyone else is considered to be malicious. TIA takes as input, any centrality measure for SSNs, a corresponding τ, as well as any set of decluttering operations.

Let V be a set of nodes. We use $\mathcal{G}$ to denote the set of all possible SSNs $G = (V, E, W)$

over V. A decluttering operation is defined as follows.

Definition 1 (Decluttering Operation) *A* decluttering operation *is an associative function* $\rho : \mathcal{G} \rightarrow \mathcal{G}$ *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* $e' \in E'$*,* $W'(e') = W(e')$*.*

Two decluttering operations ρ_1, ρ_2 can be composed. $\rho_2 \circ \rho_1(G)$ is defined as $\rho_2(\rho_1(G))$. We consider the following decluttering operations (see also Figure 6.2).

DOP(a): Remove all positive pairs of edges between benign nodes. Suppose u, v are both benign and endorse each other. In this case, we do not know whether they are really endorsing each other or whether one is blindly reciprocating endorsements. For instance, if u added v as a friend in Slashdot, v might reciprocate even if he has never read v's posts. Moreover, removing positive pairs of edges between benign nodes helps to potentially alleviate the effects of attacks B,C, and D as it also removes positive loops between pairs of malicious nodes that may have initially been misclassified as being benign, thus reducing the scores of these malicious nodes. Consider a pair of malicious users who are mistakingly classified as benign nodes and both follow attack model B. On Slashdot, this means both add each other as friends. Removing the positive edge pair between them would prevent further increase in their centrality scores. Now, consider a single malicious user following attack model C. On Slashdot, one way to trick a non-malicious user into adding a troll as a friend is to add her as a friend first, which would prompt her to add the troll as a friend too. This would create a positive edge pair between them. By removing these edges, attack model C is countered. Since attack model D is a combination of models B and C, and both are countered by this operation, so is model D.

DOP(b): Remove all negative pairs of edges between benign nodes. The removal of a negative pair of edges between two benign nodes is motivated by the fact that it might not actually mean that a user hates the other user's content. In Slashdot, a user u might add v as a foe just because v

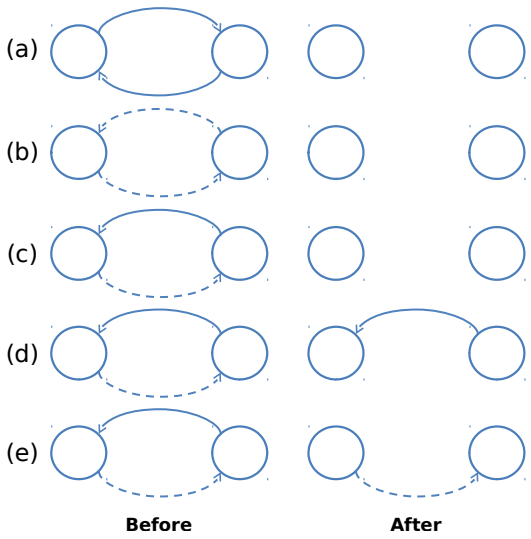


Figure 6.2: Decluttering operations in TIA. All nodes are marked benign. Bold (dashed) edges denote a positive (negative) relationship.

adds u as a foe. Removing negative edge pairs also counters attack models A and E. If a malicious user is mistakenly scored as a benign and she follows attack models A and E (a negative edge pair between her and the actual benign node), then removing this edge pair prevents a decrease in centrality of the benign user, and hence prevents increase of the malicious user's score.

DOP(c): Remove all pairs of edges between benign nodes where one edge is positive and the other is negative. The modifications related to positive-negative edge pair counters attack models C and E. Consider a malicious user, who is misidentified as a benign user, and having a negative edge to a benign user (model E) and the benign user has a positive edge to the malicious user (model C). Removing the positive edge counters attack model C, removing the negative edge counters attack model E and removing both the edges counters both the attack models in this case.

DOP(d): In a positive/negative pair between two benign nodes, remove the negative edge. Similar to the previous case.

DOP(e): In a positive/negative pair between two benign nodes, remove the positive edge. Similar rationale as for (c) above.

Example 3 *Consider the SSN in Figure 6.1 (left) with SEC as centrality measure and* $\tau = 0$*. The SEC scores are as follows:*

Node	*1*	*2*	*3*	***4***	***5***	*6*
SEC	*0.16*	*0.33*	*0.68*	*-0.30*	*-0.55*	*0.09*

Nodes 1, 2, 3, *and* 6 *can be marked "benign" according to SEC, as they have a score greater than* τ*. If we consider decluttering operations (a) and (d), we can remove the pair of positive edges between nodes* 1-3 *and* 2-3*, and the negative edge from* 6 *to* 1*. The resulting simplified network is shown in Figure 6.1 (right). Observe that the negative edge pair between nodes* 3-5 *is not removed because node* 5 *has score less than* τ*.* □

In most online social networks, the number of malicious users is a small percentage of the total number of users. Hence, the number of interactions involving malicious users is much smaller than the number of interactions involving benign users. The removal of edges proposed in *DOPs* (a)-(e) reduces the effect benign users have on the network and magnify the actions of malicious users by removing the clutter of benign-benign user interactions. Our decluttering operations also counter attack models (A)-(E). For instance, consider the situation in Example 3: due to the decluttering operations we counter attack the attack model B used by node 2 and attack models B and C (resulting in the attack model D) followed by node 3.

6.5 TIA Algorithm

In this section, we present the TIA algorithm (see pseudo-code in Figure 6.3). The algorithm takes as input, a signed social network $G = (V, E, W)$, together with any centrality measure $\mathcal{C}$ that applies to SSNs, as well as a centrality threshold τ, and a set of decluttering operations selected from our 5 decluttering operations presented above. TIA operates iteratively and proceeds as follows.

1: **Algorithm** TIA

2: **Input**: SSN $G = (V, E, W)$, centrality measure $\mathcal{C}$, a set $S = \{\rho_1, \ldots, \rho_m\}$ of decluttering operations, a threshold τ

3: **Output**: A score for nodes in V

4: **do**

5: $G' = G$

6: C $\leftarrow$ compute $\mathcal{C}$ centrality of nodes in V in graph G

7: $Benign = \{v \in V \mid centrality(C, v) \geq \tau\}$;

8: $Malicious = V - Benign$

9: $G = \rho_m \circ \cdots \circ \rho_1(G')$ %declutter graph

10: **while**$(G \neq G')$

11: Return C

Figure 6.3: Troll Identification Algorithm (TIA).

• In the first iteration, it uses the original network to compute the centrality of all nodes in V using the given centrality measure. Any node whose centrality is above the threshold τ is considered benign – all other nodes are considered malicious. It uses this initial labeling of nodes (which could be wrong) to declutter the graph using the selected decluttering operations. These operations transform the graph into a simpler graph.

• In the next iteration, we recompute the set of benign and malignant nodes using the decluttered graph and τ. The updated set of benign and malignant nodes are used in conjunction with the decluttering operations to generate an even more simplified graph. This graph is the input to the next iteration.

• All iterations follow the same pattern as above — the iterations terminate when the decluttering in Step 9 of the algorithm leads to no change.

Example 4 *Consider the toy SSN of Figure 6.1 (left). Suppose we run the* TIA *algorithm with*

NR, decluttering operations $S = \{a, b, d\}$, and a threshold $\tau = 0$. In the first iteration, shown in Figure 6.4(a), NR is computed on the original network, and nodes 2 and 3 are identified as benign users. So, the positive edge pair between these two nodes is removed resulting in the network shown in Figure 6.4(b). At this point, NR is computed again over this network and it identifies nodes 1, 3, and 6 as benign users. So, it removes the positive edge pair between nodes 1 and 3 and the negative edge from 6 to 1. This gives the network shown in Figure 6.4(c). Nodes $1, 4, 5$ and 6 are now marked as benign users by NR. Since no more decluttering operations can be further applied, the algorithm stops. At the end nodes 2 and 3 are correctly identified as trolls. In this process the same attack models that were counteracted in Example 3 are counteracted as well.

The table in Example 2 (right) shows the node rankings obtained by using TIA algorithm with the centrality measures considered in this chapter and the set of decluttering operations $S = \{a, b, d\}$.

Because the graph has been decluttered, SEC, SSR and NR are able to correctly identify both nodes 2 and 3 as trolls — something they could not do before (compare with the table in Example 2 left). □

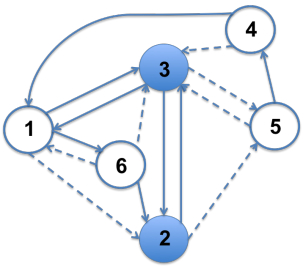
We note that TIA terminates in at most $|E|$ iterations.

6.6 Experimental Evaluation: Detection Trolls

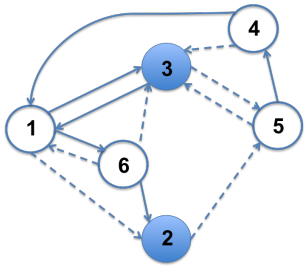
6.6.1 Slashdot Zoo Dataset Description

We tested our algorithm on a Slashdot Zoo data set.[2] This dataset is maintained by the authors of [148] and contains about 71.5K nodes and 490K edges — about 24% of the edges are

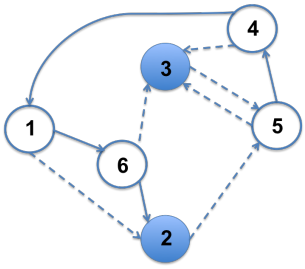
[2] `http://konect.uni-koblenz.de/networks/slashdot-zoo`. Note that this is not the same Slashdot network used in [148], as that is not available.



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



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



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

Figure 6.4: TIA algorithm iterations by using Negative Rank.

negative. 96 nodes are marked as trolls by "No More Trolls" (an administrative Slashdot account). We treat these 96 trolls as the ground truth. This is the same setting followed by [148].

We evaluated TIA's performance in conjunction with various centrality measures and with various sets of decluttering operations. For each experimental setting, we also created various subsets of the entire Slashdot Zoo network. The subsets were created by randomly removing 5%, 10%, 15%, 20% and 25% of the nodes and their corresponding edges from the entire network. For each setting, we randomly generated 50 subgraphs of the Slashdot Zoo dataset by removing the appropriate percentage of edges from the network.

6.6.2 Experiment

We implemented TIA algorithm as well as all centrality measures and decluttering operations in this chapter in about 1000 lines of Java code and ran them on a Intel Xeon @ 2.3 GHz, 24GB RAM Linux machine. We computed the score given by each centrality measure without any decluttering operations and with all possible subsets of decluttering operations (note that *DOPs* d and e together make *DOP* c, so there are 15 subsets). We use average precision and mean average precision for comparison [160] from IR (where they are used to measure goodness of document search algorithms). For systems that return a ranked sequence of documents, the measure considers the order in which the retrieved documents are presented. We use the same measure to compare the user scoring methods. Since our aim is to find malicious users, malicious and benign users are the analog of relevant and non-relevant documents, respectively.The average precision is defined as:

$$AveP = \frac{\sum_{k=1}^{n}(P(k) \times Mal(k))}{\text{Number Of Trolls}}$$

Here, *P(k)* is the precision at cut-off k (i.e. users ranked in the top k for being malicious) and *Mal(k)* is 1 if the k^{th} user is malicious, and 0 otherwise. n is the total number of users. For the experiments on the subset networks, we find the mean of the AveP values of the 50 individual sub-networks and this is reported as the Mean Average Precision (MAP).

In freaks centrality, we took the mean of the maximum and minimum value of the centrality in each round as the threshold τ for computing the sets $Malicioius$ and $Benign$ in Steps 7 and 8 of the TIA algorithm. For others, zero is the threshold. We don't include PageRank as it is not intended to identify malicious users.

Tables 6.3 and 6.4 show the average precision and the running time (in seconds), respectively, of the TIA algorithm using various subsets of decluttering operations and different SSN

centrality measures applied to the Slashdot Zoo data set. In these tables, the *None* column shows results when only the centrality measure is used with no decluttering. For some sets of decluttering operations, Signed Spectral Rank does not converge in over 100,000 iterations - due to this, Negative Rank is also not found. Cells with - depict this.

Table 6.5 shows the number of trolls found among the lowest ranked 96 users by the TIA algorithm using various subsets of decluttering operations and different SSN centrality measures. Table 6.6 shows the Mean Average Precision and the running time of the nine centrality measures and TIA algorithm using SEC with the top 2 settings that gave the best result on the original network, averaged over the 50 versions each for 95%, 90%, 85%, 80% and 75% randomly selected nodes from the Slashdot network.

Best Settings. The best results are obtained when TIA algorithm uses the Signed Eigenvector Centrality with decluttering operations *a* and *e*. In this setting, we retrieve more than twice as many trolls as Negative Rank does in the bottom 96 ranked users. The running time of this algorithm is less than 2 minutes, which is quite reasonable – and more than 27 times faster on the original network and 35-50 times faster on the sub-networks as compared to using the best centrality measure (Negative Rank) in [148].

Centrality	*None*	*a*	*b*	*c*	*d*	*e*	*a,b*	*a,c*	*a,d*	*a,e*	*b,c*	*b,d*	*b,e*	*a,b,c*	*a,b,d*	*a,b,e*
Freaks	15.07	15.22	14.92	14.77	14.77	15.22	14.92	14.77	14.77	15.22	14.46	14.46	14.92	14.46	14.46	14.92
FMF	3.13	4.6	3.13	3.12	3.13	3.13	4.2	4.35	4.33	4.64	3.13	3.12	3.13	4.01	3.96	4.25
Prestige	0.18	0.2	0.18	0.17	0.17	0.18	0.2	0.2	0.2	0.2	0.17	0.17	0.18	0.2	0.2	0.2
M-PR	1.25	0.99	1.26	1.21	1.28	1.19	0.96	0.94	0.84	1.12	1.23	1.28	1.19	0.76	0.83	0.9
SSR	10.27	-	10.44	10.05	10.34	10.03	-	-	-	-	10.17	10.46	10.16	-	-	-
NR	13.9	-	13.94	13.69	13.87	13.70	-	-	-	-	13.67	13.91	13.70	-	-	-
SEC	3.42	49.79	3.38	3.25	3.33	3.34	48.84	50.96	50.02	51.04	3.25	3.30	3.32	50.42	48.97	50.38
M-HITS	13.38	15.87	13.37	13.4	13.4	13.38	15.78	15.79	15.79	15.88	13.39	13.39	13.37	15.71	15.71	15.79
BAD	0.18	0.19	0.18	0.18	0.18	0.18	0.19	0.19	0.18	0.19	0.18	0.18	0.18	0.18	0.18	0.19

Table 6.3: Table comparing Average Precision (in %) using TIA algorithm with different centrality measures and decluttering operations on Slashdot network.

Centrality	*None*	*a*	*b*	*c*	*d*	*e*	*a,b*	*a,c*	*a,d*	*a,e*	*b,c*	*b,d*	*b,e*	*a,b,c*	*a,b,d*	*a,b,e*
Freaks	0.25	2.89	2.48	2.02	1.89	2.69	4.57	4.82	4.72	4.62	4.53	4.55	4.53	6.05	6.98	6.99
FMF	0.41	2.31	1.74	2.12	2.2	2.85	3.4	2.99	2.99	2.98	4.6	3.09	3.07	2.84	2.83	2.83
Prestige	0.59	2.58	2.0	2.36	2.29	2.26	3.05	3.13	3.16	3.14	4.66	3.36	3.34	2.88	3.01	3.09
M-PR	13.8k	65.6k	38.4k	39.2k	26.4k	50.6k	73.1k	63.8k	59.9k	67.5k	39.7k	49.7k	51.9k	73.3k	78.6k	76.6k
SSR	2.5k	-	7.5k	10.3k	7.6k	7.6k	-	-	-	-	9.9k	6.1k	7.5k	-	-	-
NR	3.2k	-	8.5k	8.7k	8.4k	8.6k	-	-	-	-	8.1k	8.5k	8.5k	-	-	-
SEC	8.74	121.93	41.31	49.68	52.44	49.07	107.96	118.21	123.96	114.49	56.48	58.51	57.37	87.75	88.12	88.64
M-HITS	28.58	106.99	89.73	58.02	58.29	58.7	114.13	101.89	105.13	100.84	90.21	82.71	89.97	105.32	103.26	103.77
BAD	35.72	110.52	108.67	71.38	105.2	71.74	101.79	100.08	99.15	99.16	108.58	110.49	111.86	104.76	105.29	100.74

Table 6.4: Table comparing running time (in sec.) using TIA algorithm with different centrality measures and decluttering operations on Slashdot network.

Centrality	*None*	*a*	*b*	*c*	*d*	*e*	*a,b*	*a,c*	*a,d*	*a,e*	*b,c*	*b,d*	*b,e*	*a,b,c*	*a,b,d*	*a,b,e*
Freaks	17	17	16	16	16	17	16	16	16	17	15	15	16	15	15	16
FMF	10	10	10	10	10	10	9	10	10	10	10	10	10	9	10	9
Prestige	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-PR	6	6	5	5	6	5	6	6	5	7	5	5	5	5	6	6
SSR	19	-	20	19	19	19	-	-	-	-	20	20	20	-	-	-
NR	24	-	24	23	24	23	-	-	-	-	23	25	23	-	-	-
SEC	7	50	6	5	6	6	50	51	51	51	5	5	6	51	50	50
M-HITS	16	19	16	17	17	16	19	19	19	19	17	17	16	19	19	19
BAD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6.5: Number of trolls found among the lowest ranked 96 users (the number of trolls in Slasdot is 96) by using TIA algorithm with different centrality measures and decluttering operations on Slashdot network.

	95%		90%		85%		80%		75%	
Measure	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 6.6: Table showing Mean Average Precision (MAP) and runtime of the five centrality measures and TIA with SEC and the top 2 decluttering operations, averaged over 50 different versions each for 95%, 90%, 85%, 80% and 75% randomly selected nodes from the Slashdot network.

6.7 Conclusion

The problem of detecting trolls in online environments like Slashdot and other signed social networks is increasingly important as open source, collaboratively edited information becomes used more widely. Ensuring the integrity of this information is important for users while posing a technical challenge as many entities have strong incentives to compromise the accuracy of such information.

In this chapter, we have shown an unsupervised graph mining algorithm that can significantly improve on detection of trolls on Slashdot Zoo using a suite of decluttering operations that simplify a signed social network by removing confusing or irrelevant edges from the network. We proposed the TIA algorithm that takes any centrality measure and any set of decluttering operations as input parameters, and uses them to iteratively identify the trolls in the social network. Using the standard Average Precision measure to capture accuracy of our TIA algorithm, we show that TIA using Signed Eigenvector Centrality and decluttering operations *a* and *e* gives us the best result of 51.04%, significantly exceeding the 15.07% when no decluttering operations are performed. Compared to the best existing results on troll detection in Slashdot [3], our algorithm runs 27 times faster on the original network and 35-50 times faster on the sub-networks. Moreover it is able to retrieve about twice as many trolls with a Mean Average Precision which is more than three times as good as [148]. *The final message is simple: decluttering SSNs helps expose a clearer picture to our TIA algorithm which enables it to achieve much higher precision as well as identify far more trolls — all while running faster.*

In the next chapter, we look at a supervised graph mining algorithm to detect malicious actors on nuclear networks.

Chapter 7

Predicting Trustworthy Users in Rating Platforms

This is the second graph-based algorithm we will look at, particularly in the context of detecting fraudulent users in rating platforms. This algorithm iteratively evaluates the fairness of each user, the reliability of each review, and the goodness of each product, all at the same time. It also combines it with behavioral attributes to penalize suspicious behavior. Together this algorithm, called the FairJudge algorithm, outperforms several existing algorithms in identifying fraudulent users in several rating platforms.

7.1 Introduction

Consumer generated ratings are now an essential part of several platforms. For instance, users on Yelp and TripAdvisor rate restaurants, hotels, attractions, and various types of services. Every major online marketplace (eBay, Amazon, Flipkart) uses online ratings as a way of recognizing good products and rewarding honest/good behavior by vendors. Because buyers frequently look at reviews before buying a product or using a vendor, there is a huge incentive for unethical

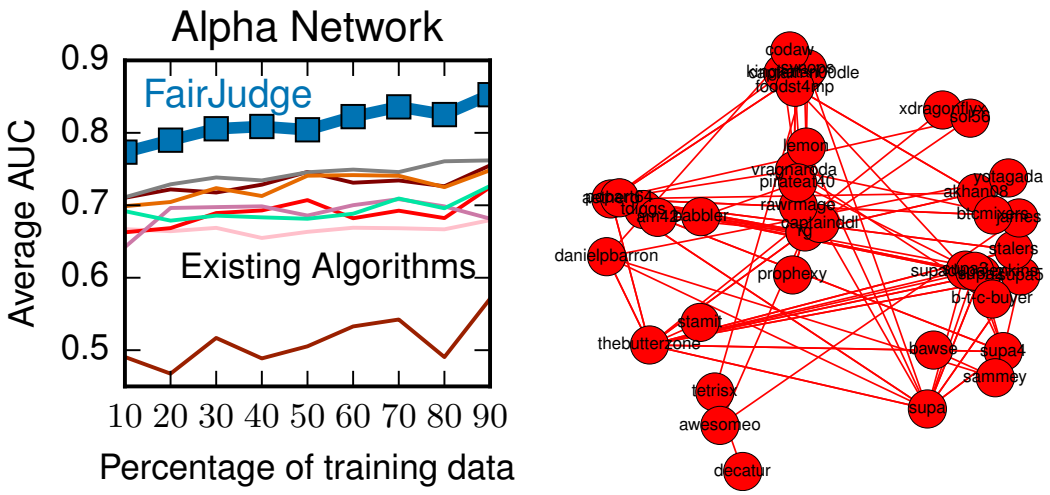
entities to give fraudulent ratings to compromise the integrity of these consumer generated rating systems [39–42]. The goal of this chapter is to identify such unfair users.

In this chapter, we present three novel metrics to quantify the trustworthiness of users and reviews, and the quality of products, building on our prior work [43]. We model user-to-item ratings with timestamps as a bipartite graph. For instance, on an online marketplace such as Amazon, a user u rates a product p with a rating (u, p). Each user has an intrinsic level of fairness $F(u)$, each product p has an intrinsic goodness $G(p)$ (measuring its quality), and each rating (u, p) has an intrinsic reliability $R(u, p)$. Intuitively, a fair user should give ratings that are close to the goodness of the product, and good products should get highly positive reliable ratings. Clearly, these $F(u), G(p), R(u, p)$ metrics are all inter-related, and we propose three mutually-recursive equations to model them.

However, it is not possible to predict the true trustworthiness of users that have only given a few ratings. For example, users with only a few ratings, all of which are highly accurate, can be a fraudulent shill account building initial reputation [42] or it can be a genuine user. Similarly, the true quality of products that have received only a few ratings is also uncertain [44,45]. We propose a Bayesian solution to to address these *cold start problems* in our formulation by incorporating priors to users' fairness and products' goodness scores.

Additionally, the rating behavior is often very indicative of their nature. For instance, unusually rapid or regular behavior has been associated with fraudulent entities, such as fake accounts, sybils and bots [46–49]. Similarly, unusually bursty ratings received by a product may be indicative of fake reviews [50]. Therefore, we propose a Bayesian technique to incorporate users' and products' rating behavior in the formulation, by penalizing unusual behavior [48].

Combining the network, cold start treatment and behavioral properties together, we present the FairJudge formulation and an iterative algorithm to find the fairness, goodness and reliabil-



(a) (b)

Figure 7.1: (a) The proposed algorithm, FairJudge, consistently performs the best, by having the highest AUC, in predicting fair and unfair users with varying percentage of training labels. (b) FairJudge discovered a bot-net of 40 confirmed shills of one user, rating each other positively.

ity scores of all entities together. We prove that FairJudge has linear time complexity and it is guaranteed to converge in a bounded number of iterations.

How well does FairJudge work? We conduct extensive experiments using 5 real-world data sets – two Bitcoin user trust networks, Epinions, Amazon and Flipkart, India's biggest online marketplace. With the help of five experiments, we show that FairJudge outperforms several existing methods [44, 48, 51–56] in predicting fraudulent users. Specifically, in an unsupervised setting, FairJudge has the best or second best average precision in nine out of ten cases. Across two supervised settings, FairJudge has the highest AUC ≥ 0.85 across all five datasets. It consistently performs the best as the percentage of training data is varied between 10 and 90, as shown for the Alpha network in Figure 7.1(a). Further, we experimentally show that both the cold start treatment and behavior properties improve the performance of FairJudge algorithm, and incorporating both together performs even better.

FairJudge is practically very useful. We reported the 100 most unfair users as predicted by

FairJudge in the Flipkart online marketplace. Review fraud investigators at Flipkart studied our recommendations and confirmed 80 of them were unfair, presenting a validation of the utility of FairJudge in identifying real-world review fraud. In fact, *FairJudge is already being deployed at Flipkart.* On the Bitcoin Alpha network, using FairJudge, we discovered a botnet-like structure of 40 unfair users that rate each other positively (Figure 7.1(b)), which are confirmed shills of a single account.

Overall, this chapter makes the following contributions:

• **Algorithm**: We propose three novel metrics called fairness, goodness and reliability to rank users, products and ratings, respectively. We propose Bayesian approaches to address cold start problems and incorporate behavioral properties. We propose the FairJudge algorithm to iteratively compute these metrics.

• **Theoretical guarantees**: We prove that FairJudge algorithm is guaranteed to converge in a bounded number of iterations, and it has linear time complexity.

• **Effectiveness**: We show that FairJudge outperforms nine existing algorithms in identifying fair and unfair users, conducted via five experiments on five rating networks.

7.2 Related Work

Existing works in rating fraud detection can be categorized into network-based and behavior-based algorithms:

Network-based fraud detection algorithms are based on iterative learning, belief propagation, and node ranking techniques. Similar to our proposed FairJudge algorithm, [52, 54, 95] develop iterative algorithms that jointly assign scores in the rating networks based on consensus of ratings - [54] scores each user, review and product, and [52] scores each user and product. FraudEagle [51] is a belief propagation model to rank users, which assumes fraudsters rate good products poorly

	BIRDNEST [48]	Trustiness [54]	BAD [52]	FraudEagle [51]	SpEagle [53]	[44, 55, 56]	FairJudge
Uses network information	✓	✓	✓	✓	✓		✓
Uses behavior properties	✓				✓	✓	✓
Prior free			✓			✓	✓
Theoretical Guarantees			✓				✓

Table 7.1: Our algorithm FairJudge satisfies all desirable properties.

and bad products positively, and vice-versa for honest users. Random-walk based algorithms have been developed to detect trolls [94] and link farming from collusion on Twitter [87]. [86, 95, 161] identify group of fraudsters based on local neighborhood of the users. A *survey* on network-based fraud detection can be found in [97].

Behavioral fraud detection algorithms are often feature-based. Consensus based features have been proposed in [44, 55] – our proposed goodness metric is also inspired by consensus or 'wisdom of crowds'. Commonly used features are derived from timestamps [45, 50, 82] and review text [162–164]. SpEagle [53] extends FraudEagle [51] to incorporate behavior features. BIRDNEST [48] creates a Bayesian model to estimate the belief of each user's deviation in rating behavior from global expected behavior. [49, 86, 96] study coordinated spam behavior of multiple users. A *survey* on behavior based algorithms can be found in [165].

Our FairJudge algorithm combines both network and behavior based algorithms, along with Bayesian solution to cold start problems. Since review text is not always available, for e.g. in a subset of our datasets, we only focus on ratings and timestamps. FairJudge provides theoretical guarantees and does not require any user inputs. Table 7.1 compares FairJudge to the closest existing algorithms, which do not always satisfy all the desirable properties.

7.3 FairJudge Formulation

In this section, we present the FairJudge algorithm that jointly models the rating network and behavioral properties. We first present our three novel metrics — Fairness, Reliability and Goodness — which measure intrinsic properties of users, ratings and products, respectively, building on our prior work [43]. We show how to incorporate Bayesian priors to address user and product cold start problems, and how to incorporate behavioral features of the users and products. We then prove that our algorithm have several desirable theoretical guarantees.

Prerequisites. We model the rating network as a bipartite network where user u gives a rating (u, p) to product p. Let the rating score be represented as `score(u,p)`. Let $\mathcal{U}, \mathcal{R}$ and $\mathcal{P}$ represent the set of all users, ratings and products, respectively, in a given bipartite network. We assume that all rating scores are scaled to be between -1 and +1, i.e. `score(u,p)` $\in [-1, 1] \forall (u, p) \in \mathcal{R}$. Let, `Out(u)` be the set of ratings given by user u and `In(p)` be the set of ratings received by product p. So, |`Out(u)`| and |`In(p)`| represents their respective counts.

7.3.1 Fairness, Goodness and Reliability

Users, ratings and products have the following characteristics:

• Users vary in ***fairness***. Fair users rate products without bias, i.e. they give high scores to high quality products, and low scores to bad products. On the other hand, users who frequently deviate from the above behavior are 'unfair'. For example, fraudulent users often create multiple accounts to boost ratings of unpopular products and bad-mouth good products of their competitors [42, 49]. Hence, these fraudsters should have low fairness scores. The fairness $F(u)$ of a user u lies in the $[0, 1]$ interval $\forall u \in \mathcal{U}$. 0 denotes a 100% untrustworthy user, while 1 denotes a 100% trustworthy user.

• Products vary in terms of their quality, which we measure by a metric called ***goodness***. The quality of a product determines how it would be rated by a fair user. Intuitively, a good product would get several high positive ratings from fair users, and a bad product would receive high negative ratings from fair users. The goodness $G(p)$ of a product p ranges from -1 (a very low quality product) to $+1$ (a very high quality product) $\forall p \in \mathcal{P}$.

• Finally, ratings vary in terms of ***reliability***. This measure reflects how trustworthy the specific rating is. The reliability $R(u, p)$ of a rating (u, p) ranges from 0 (an untrustworthy rating) to 1 (a trustworthy rating) $\forall (u, p) \in \mathcal{R}$

The reader may wonder: *isn't the rating reliability, identical to the user's fairness?* The answer is 'no'. Consider Figure 7.2, where we show the rating reliability distribution of the top 1000 fair and top 1000 unfair users in the Flipkart network, as identified by our FairJudge algorithm (explained later). Notice that, while most ratings by fair users have high reliability, some of their ratings have low reliability, indicating personal opinions that disagree with majority (see green arrow). Conversely, unfair users give some high reliability ratings (red arrow), probably to camouflage themselves as fair users. Thus, having reliability as a rating-specific metric allows us to more accurately characterize this distribution.

Given a bipartite user-product graph, we do not know the values of these three metrics for any user, product or rating. Clearly, these scores are mutually interdependent. Let us look at an example.

Example 5 (Running Example) *Figure 7.3 shows a simple example in which there are 3 products, P_1 to P_3, and 6 users, U_A to U_F. Each review is denoted as an edge from a user to a product, with rating score between -1 and $+1$. Note that this is a rescaled version of the traditional 5-star rating scale where a 1, 2, 3, 4 and 5 star corresponds to $-1, -0.5, 0, +0.5$ and $+1$ respectively.*

One can immediately see that U_F's ratings are inconsistent with those of the U_A, U_B, U_C,

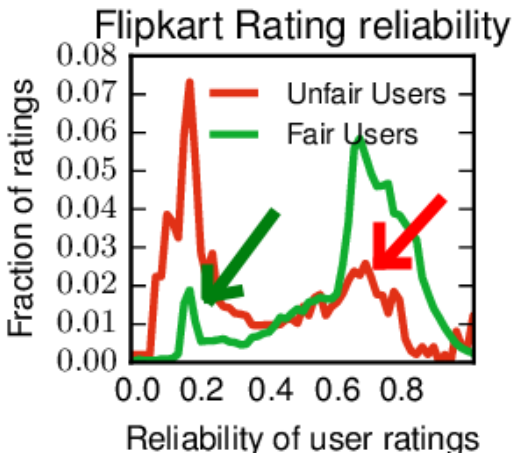


Figure 7.2: While most ratings of fair users have high reliability, some ratings also have low reliability (green arrow). Conversely, unfair users also give some highly reliability ratings (red arrow), but most of their ratings have low reliability.

U_D and U_E. U_F gives poor ratings to P_1 and P_2 which the others all agree are very good by consensus. U_F also gives a high rating for P_3 which the others all agree is very bad. We will use this example to motivate our formal definitions below.

Fairness of users. Intuitively, fair users are those who give reliable ratings, and unfair users mostly give unreliable ratings. So we simply define a user's fairness score to be the average reliability score of its ratings:

$$F(u) = \frac{\sum_{(u,p) \in Out(u)} R(u,p)}{|Out(u)|} \tag{7.4}$$

Other definitions such as temporally weighted average can be developed, but we use the above for simplicity.

Goodness of products. When a product receives rating scores via ratings with different reliability, clearly, more importance should be given to ratings that have higher reliability. Hence, to estimate a product's goodness, we weight the ratings by their reliability scores, giving higher weight to

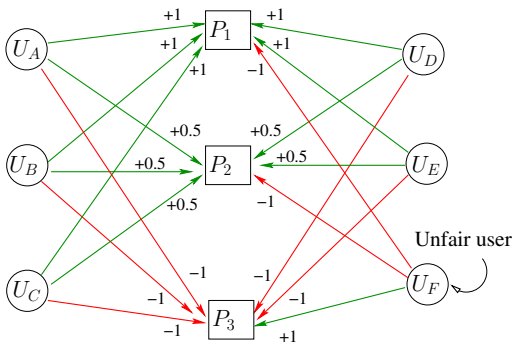


Figure 7.3: Toy example showing products (P_1, P_2, P_3), users (U_A, U_B, U_C, U_D, U_E and U_F), and rating scores provided by the users to the products. User U_F always disagrees with the consensus, so U_F is unfair.

reliable ratings and little importance to low reliability ratings:

$$G(p) = \frac{\sum_{(\mathtt{u},\mathtt{p}) \in \mathtt{In}(\mathtt{p})} R(u,p) \cdot \mathtt{score}(\mathtt{u},\mathtt{p})}{|\mathtt{In}(\mathtt{p})|} \tag{7.5}$$

Returning to our *running example*, the ratings given by users U_A, U_B, U_C, U_D, U_E and U_F to product P_1 are $+1, +1, +1, +1, +1$ and -1, respectively. So we have:

$$G(P_1) = \frac{1}{6}(R(U_A, P_1) + R(U_B, P_1) + R(U_C, P_1) + R(U_D, P_1) + R(U_E, P_1) - R(U_F, P_1))$$

Reliability of ratings. A rating (u, p) should be considered reliable if (i) it is given by a generally fair user u, and (ii) its score is close to the goodness score of product p. The first condition makes sure that ratings by fair users are trusted more, in lieu of the user's established reputation. The second condition leverages 'wisdom of crowds', by making sure that ratings that deviate from p's goodness have low reliability. This deviation is measured as the normalized absolute difference, $|\mathtt{score}(\mathtt{u},\mathtt{p}) - G(p)|/2$. Together:

$$R(u,p) = \frac{1}{2}(F(u) + (1 - \frac{|\mathtt{score}(\mathtt{u},\mathtt{p}) - G(p)|}{2})) \tag{7.6}$$

$$\text{Fairness of user } u,\ F(u) = \frac{0.5 \cdot \alpha_1 + \alpha_2 \cdot IBIRDNEST_{IRTD_U}(u) + \sum_{(u,p)\in \texttt{Out(u)}} R(u,p)}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|} \tag{7.1}$$

$$\text{Reliability of rating } (u,p),\ R(u,p) = \frac{1}{2}(F(u) + (1 - \frac{|\texttt{score(u,p)} - G(p)|}{2})) \tag{7.2}$$

$$\text{Goodness of product } p,\ G(p) = \frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum_{(u,p)\in \texttt{In(p)}} R(u,p) \cdot \texttt{score(u,p)}}{\beta_1 + \beta_2 + |\texttt{In(p)}|} \tag{7.3}$$

Figure 7.4: This is the set of mutually recursive definitions of fairness, reliability and goodness for the proposed FairJudge algorithm. The yellow shaded part addresses the cold start problems and gray shaded part incorporates the behavioral properties.

In our *running example*, for the rating by user U_F to P_1:

$$R(U_F, P_1) = \frac{1}{2}\left(F(U_F) + (1 - \frac{|-1 - G(P_1)|}{2})\right)$$

Similar equations can be associated with every edge (rating) in the graph of Figure 7.3.

7.3.2 Addressing Cold Start Problems

If a user u has given only a few ratings, we have very little information about his true behavior. Say all of u's ratings are very accurate – it is hard to tell whether it is a fraudster that is camouflaging and building reputation by giving genuine ratings [42], or it is actually a benign user. Conversely, if all of u's ratings are very deviant, it is hard to tell whether the user is a fraudulent shill account [44, 55], or simply a normal user whose rating behavior is unusual at first but stabilizes in the long run [48]. Due to the lack of sufficient information about the ratings given by the user, little can be said about his fairness. Similarly, for products that have only been rated a few times, it is hard to accurately determine their true quality, as they may be targets of fraud [44, 45]. This uncertainty due to insufficient information of less active users and products is the *cold start problem*.

We solve this issue by assigning Bayesian priors to each user's fairness score as follows:

$$F(u) = \frac{0.5 \cdot \alpha + \sum_{(u,p)\in \texttt{Out(u)}} R(u,p)}{\alpha + |\texttt{Out(u)}|} \tag{7.7}$$

Here, α is a non-negative integer constant, which is the relative importance of the prior compared to the rating reliability – the lower (higher) the value of α, the more (less, resp.) the fairness score depends on the reliability of the ratings. The 0.5 score is the default global prior belief of all users' fairness, which is the midpoint of the fairness range $[0, 1]$. If a user gives only a few ratings, then the fairness score of the user is close to the default score 0.5. The more number of ratings the user gives, the more the fairness score moves towards the user's rating reliabilities.

This way shills with few ratings have little effect on product scores.

Similarly, the Bayesian prior in product's goodness score is incorporated as:

$$G(p) = \frac{\sum_{(u,p)\in \texttt{In(p)}} R(u,p) \cdot \texttt{score(u,p)}}{\beta + |\texttt{In(p)}|} \tag{7.8}$$

Again, β is a non-negative integer constant. The prior belief of product's goodness is set to 0 which is the midpoint of the goodness range $[-1, 1]$, and so $0 \cdot \beta = 0$ does not appear in the numerator. We will explain how the values of α and β are set in Section 7.4.

7.3.3 Incorporating Behavioral Properties

Rating scores alone are not sufficient to efficiently estimate the fairness, goodness and reliability values. The behavior of the users and products is also an important aspect to be considered. As an example, fraudsters have been known to give several ratings in a very short timespan, or regularly at fixed intervals which indicates bot-like behavior [47–49]. Even though the ratings themselves may be very accurate, the unusually rapid behavior of the user is suspicious. Benign users, on the other hand, have a more spread out rating behavior as they lack regularity [47]. In addition, products that receive an unusually high number of ratings for a very short time period may have bought fake ratings [45, 50]. Therefore, behavioral properties of the ratings that a user gives or a product receives are indicative of their true nature.

Here, we show a Bayesian technique to incorporate the rating behavioral properties of users and products. We focus on temporal rating behavior as the behavioral property for the rest of this chapter, but our method can be used with any additional set of properties. A user u's temporal rating behavior is represented as the time difference between its consecutive ratings, *inter-rating time distribution* $IRTD_U(u)$. To model the behavior, we use a recent algorithm called BIRDNEST [48], which calculates a Bayesian estimate of how much user u's $IRTD_U(u)$ deviates from the global population of all users' behavior, $IRTD_U(u')\forall u' \in \mathcal{U}$.

This estimate is the BIRDNEST score of user u, $BIRDNEST_{IRTD_U}(u) \in [0,1]$. We use $IBIRDNEST_{IRTD_U}(u) = 1 - BIRDNEST_{IRTD_U}(u)$, as user u's behavioral normality score. When

$IBIRDNEST_{IRTD_U}(u)$ = 1, it means the user's behavior is absolutely normal, and 0 means totally abnormal. Similarly, for products, $IBIRDNEST_{IRTD_P}(p)$ is calculated from the consecutive ratings' time differences $IRTD_P(p)$ that product p gets, with respect to the global $IRTD_P(p')\forall p' \in \mathcal{P}$.

As in the case of cold start, we adopt a Bayesian approach to incorporate the behavioral properties in the fairness and goodness equations 7.7 and 7.8. The $IBIRDNEST_{IRTD_U}(u)$ score is treated as user u's behavioral Bayesian prior. As opposed to global priors (0.5 and 0) in the cold start case, this prior is user-dependent and may be different for different users. The resulting equation is given in Equation 7.1. Note that now there are two non-negative integer constants, α_1 and α_2, indicating the relative importance of the cold start and behavioral components, respectively, to the network component. The Bayesian prior for products is incorporated similarly to give the final goodness equation in Equation 7.3.

Overall, the FairJudge formulation represented in Figure 7.4 is the set of three equations that define the fairness, goodness and reliability scores in terms of each other. It combines the network, cold start treatment and behavioral properties together.

7.4 The FairJudge Algorithm

Having formulated the fairness, goodness and reliability metrics in Section 7.3, we present the FairJudge algorithm in Algorithm 1 to calculate their values for all users, products and ratings. The algorithm is iterative, so let F^t, G^t and R^t denote the fairness, goodness and reliability score at the end of iteration t. Given the rating network and non-negative integers $\alpha_1, \alpha_2, \beta_1$ and β_2, we

first initialize all scores to the highest value 1 (see line 3).[1] Then we iteratively update the scores using Equations (7.3), (7.2) and (7.1) until convergence (see lines 6-12). Convergence occurs when all scores change minimally (see line 12). ϵ is the acceptable error bound, which is set to a very small value, say 10^{-6}.

But how do we set the values of $\alpha_1, \alpha_2, \beta_1$ and β_2? In an ***unsupervised*** scenario, it is not possible to find the best combination of these values apriori. Therefore, the algorithm is run for several combinations of $\alpha_1, \alpha_2, \beta_1$ and β_2 as inputs, and the final scores of a user across all these runs are averaged to get the final FairJudge score of the user. Formally, let $\mathcal{C}$ be the set of all parameter combinations $\{\alpha_1, \alpha_2, \beta_1, \beta_2\}$, and $F(u|\alpha_1, \alpha_2, \beta_1, \beta_2)$ be the fairness score of user u after running Algorithm 1 with $\alpha_1, \alpha_2, \beta_1$ and β_2 as input. So the final fairness score of user u is $F(u) = \frac{\sum_{(\alpha_1,\alpha_2,\beta_1,\beta_2)\in\mathcal{C}} F(u|\alpha_1,\alpha_2,\beta_1,\beta_2)}{|\mathcal{C}|}$. Similarly, $G(p) = \frac{\sum_{(\alpha_1,\alpha_2,\beta_1,\beta_2)\in\mathcal{C}} G(p|\alpha_1,\alpha_2,\beta_1,\beta_2)}{|\mathcal{C}|}$ and $R(u,p) = \frac{\sum_{(\alpha_1,\alpha_2,\beta_1,\beta_2)\in\mathcal{C}} R((u,p)|\alpha_1,\alpha_2,\beta_1,\beta_2)}{|\mathcal{C}|}$. In our experiments, we varied all (integer) parameters from 0 to 5, i.e. $0 \leq \alpha_1, \alpha_2, \beta_1, \beta_2 \leq 5$, giving 6x6x6x6 = 1296 combinations. So, altogether, scores from 1296 different runs were averaged to get the final *unsupervised* FairJudge scores. This final score is used for ranking the users.

In a ***supervised*** scenario, it is indeed possible to learn the relative importance of parameters. We represent each user u as a feature vector of its fairness scores across several runs, i.e. $F(u|\alpha_1, \alpha_2, \beta_1, \beta_2), \forall(\alpha_1, \alpha_2, \beta_1, \beta_2) \in \mathcal{C}$ are the features for user u. Given a set of fraudulent and benign user labels, a random forest classifier is trained that learns the appropriate weights to be given to each score. The higher the weight, the more important the particular combination of parameter values is. The learned classifier's prediction labels are then used as the *supervised* FairJudge output.

Example 6 (Running Example) *Let us revisit our running example. Let,* $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 =$

[1] Random initialization gives similar results.

Algorithm 1 FairJudge Algorithm

1: **Input**: Rating network $(\mathcal{U}, \mathcal{R}, \mathcal{P})$, $\alpha_1, \alpha_2, \beta_1, \beta_2$

2: **Output**: Fairness, Reliability and Goodness scores, given $\alpha_1, \alpha_2, \beta_1$ and β_2

3: Initialize $F^0(u) = 1, R^0(u,p) = 1$ and $G^0(p) = 1, \forall u \in \mathcal{U}, (u,p) \in \mathcal{R}, p \in \mathcal{P}$.

4: Calculate $IBIRDNEST_{IRTD_U}(u)\ \forall u \in \mathcal{U}$ and $IBIRDNEST_{IRTD_P}(p)\ \forall p \in \mathcal{P}$.

5: $t = 0$

6: **do**

7: $t = t + 1$

8: Update goodness of products using Equation 7.3: $\forall p \in \mathcal{P}$,

$G^t(p) = \frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum_{(u,p)\in \mathtt{In(p)}} R^{t-1}(u,p).\mathtt{score(u,p)}}{\beta_1 + \beta_2 + |\mathtt{In(p)}|}$.

9: Update reliability of ratings using Equation 7.2 $\forall (u,p) \in \mathcal{R}$,

$R^t(u,p) = \frac{1}{2}(F^{t-1}(u) + (1 - \frac{|\mathtt{score(u,p)} - G^t(p)|}{2}))$.

10: Update fairness of users using Equation 7.1 $\forall u \in \mathcal{U}$,

$F^t(u) = \frac{0.5 \cdot \alpha_1 + \alpha_2 \cdot IBIRDNEST_{IRTD_U}(u) + \sum_{(u,p)\in \mathtt{Out(u)}} R^t(u,p)}{\alpha_1 + \alpha_2 + |\mathtt{Out(u)}|}$

11: error = $\max(\max_{u\in\mathcal{U}} |F^t(u) - F^{t-1}(u)|, \max_{(u,p)\in\mathcal{R}}$

$|R^t(u,p) - R^{t-1}(u,p)|, \max_{p\in\mathcal{P}} |G^t(p) - G^{t-1}(p)|)$

12: **while** $error > \epsilon$

13: **Return** $F^{t+1}(u), R^{t+1}(u,p), G^{t+1}(p), \forall u \in \mathcal{U}, (u,p) \in \mathcal{R}, p \in \mathcal{P}$

0. *We initialize all fairness, goodness and reliability scores to 1 (line 3 in Algorithm 1. Consider the first iteration of the loop (i.e. when t is set to 1 in line 7). In line 8, we must update the goodness of all products. Let us start with product P_1. Its goodness $G^0(P_1)$ was 1, but this gets updated to*

$$G^1(P_1) \quad = \quad \frac{-1(1) + 1(1) + 1(1) + 1(1) + 1(1) + 1(1)}{6} = 0.67.$$

We see that the goodness of P_1 has dropped because of U_A's poor rating. The following table shows how the fairness and goodness values change over iterations (we omit reliability for brevity):

Node	Property	Fairness/Goodness in iterations				
		0	*1*	*2*	*5*	*9 (final)*
P_1	$G(P_1)$	*1*	*0.67*	*0.67*	*0.67*	*0.68*
P_2	$G(P_2)$	*1*	*0.25*	*0.28*	*0.31*	*0.32*
P_3	$G(P_3)$	*1*	*-0.67*	*-0.67*	*-0.67*	*-0.68*
U_A - U_E	$F(U_A)$ - $F(U_E)$	*1*	*0.92*	*0.89*	*0.86*	*0.86*
U_F	$F(U_F)$	*1*	*0.62*	*0.43*	*0.24*	*0.22*

By symmetry, nodes U_A to U_E have the same fairness values throughout. After convergence, U_F has low fairness score, while U_A to U_E have close to perfect scores. Confirming our intuition, the algorithm quickly learns that U_F is an unfair user as all of its ratings disagree with the rest of the users. Hence, the algorithm then downweighs U_F's ratings in its estimation of the products' goodness, raising the score of P_1 and P_2 as they deserve.

7.4.1 Theoretical Guarantees of FairJudge

Here we present the theoretical properties of FairJudge. Let, $F^{\infty}(u), G^{\infty}(p)$ and $R^{\infty}(u,p)$ be the final scores after convergence, for some input $\alpha_1, \alpha_2, \beta_1, \beta_2$.

Lemma 1 (Lemma 1) *The difference between a product p's final goodness score and its score after the first iteration is at most 1, i.e. $|G^{\infty}(p) - G^{1}(p)| \leq 1$. Similarly, $|R^{\infty}(u,p) - R^{1}(u,p)| \leq 3/4$ and $|F^{\infty}(u) - F^{t}(u)| \leq 3/4$.*

We prove this lemma formally in the Appendix 9.

Theorem 1 (Convergence Theorem) *The difference during iterations is bounded as $|R^{\infty}(u,p) - R^{t}(u,p)| \leq (\frac{3}{4})^t, \forall (u,p) \in \mathcal{R}$. As t increases, the difference decreases and $R^{t}(u,p)$ converges to $R^{\infty}(u,p)$. Similarly, $|F^{\infty}(u) - F^{t}(u)| \leq (\frac{3}{4})^t, \forall u \in \mathcal{U}$ and $|G^{\infty}(p) - G^{t}(p)| \leq (\frac{3}{4})^{(t-1)}, \forall p \in \mathcal{P}$.*

We prove this theorem formally in the Appendix 9. As the algorithm converges for all $\alpha_1, \alpha_2, \beta_1, \beta_2$, the entire algorithm is guaranteed to converge.

Corollary 1 (Iterations till Convergence) *The number of iterations needed to reach convergence is at most* $2 + \lceil \frac{\log(\epsilon/2)}{\log(3/4)} \rceil$. *In other words, treating* ϵ *as constant, the number of iterations needed to reach convergence is bounded by a constant.*

Again, the proof is shown in the Appendix 9.

Linear algorithmic time complexity. In each iteration, the algorithm updates goodness, reliability and fairness scores of each product, rating and user, respectively. Each of these updates takes constant time. So, the complexity of each iteration is $\mathcal{O}(|E|+|V|) = \mathcal{O}(|E|)$. By Corollary 2, the algorithm converges in a constant number of iterations. Hence the time complexity is $\mathcal{O}(k|E|)$, which is linear in the number of edges, and k is a constant equal to the product of the number of iterations till convergence and the number of runs of the algorithm.

7.5 Experimental Evaluation: Detecting Fraudulent Users

In this section, we present the results of the proposed FairJudge algorithm to identify fair and unfair users. We conduct extensive experiments on five different rating networks and show five major results:

(i) We compare FairJudge algorithm with five recent algorithms to predict benign and malicious users in an unsupervised setting. We show that FairJudge performs the best or second best in nine out of ten cases.

(ii) We show that FairJudge outperforms nine algorithms across all datasets, when training data is available.

(iii) We show that FairJudge is robust to the percentage of training data available, and consistently

Network	# Users (% unfair, fair)	# Products	# Edges
OTC	4,814 (3.7%, 2.8%)	5,858	35,592
Alpha	3,286 (3.1%, 4.2%)	3,754	24,186
Amazon	256,059 (0.09%, 0.92%)	74,258	560,804
Flipkart	1,100,000 (-, -)	550,000	3,300,000
Epinions	120,486 (0.84%, 7.7%)	132,585	4,835,208

Table 7.2: Five rating networks used for evaluation.

performs the best.

(iv) We show that both cold start treatment and behavior properties improve the performance of FairJudge algorithm, and incorporating both of them together perform the best.

(v) We show the linear running time of FairJudge.

The FairJudge algorithm is already being deployed at Flipkart.

7.5.1 Datasets: Rating Networks

We use the following five datasets. Table 7.2 has their properties. All ratings are rescaled between -1 and +1.

• **Flipkart** is India's biggest online marketplace where users rate products. The ground truth labels are generated by review fraud investigators in Flipkart, who look at various properties of the user, rating and the product being rated.

• **Epinions** network has two components – user-to-post rating network and user-to-user trust network [166]. Algorithms are run on the rating network and ground truth is defined using the trust network – a user is defined as trustworthy if its total trust rating is $\geq +10$, and unfair if ≤ -10.

• **Amazon** is a user-to-product rating network [167]. The ground truth is defined using helpfulness votes, which is indicative of malicious behavior [163] – users with at least 50 votes are trustworthy

	Unfair user prediction					Fair user prediction				
	OTC	Alpha	Amazon	Epinions	Flipkart	OTC	Alpha	Amazon	Epinions	Flipkart
FraudEagle	93.67	86.08	47.21	*nc*	*nc*	86.94	71.99	96.88	*nc*	*nc*
BAD	79.75	63.29	55.92	58.31	79.96	77.41	68.31	97.19	97.09	38.07
SpEagle	74.40	68.42	12.16	*nc*	*nc*	80.91	82.23	93.42	*nc*	*nc*
BIRDNEST	61.89	53.46	19.09	37.08	85.71	46.11	77.18	93.32	98.53	62.47
Trustiness	74.11	49.40	40.05	*nc*	*nc*	84.09	78.19	97.33	*nc*	*nc*
FairJudge	86.03	76.43	56.18	63.43	57.14	90.80	86.16	97.35	99.35	39.27

Table 7.3: Unsupervised Predictions: The table shows the Average Precision values of all algorithms in unsupervised prediction of unfair and fair users across five datasets. The best algorithm in each column is colored blue and second best is gray. Overall, FairJudge performs the best or second best in 9 of the 10 cases. *nc* indicates 'no convergence'.

(helpful) if the proportion of helpful-to-total votes is ≥ 0.75, and untrustworthy if ≤ 0.25.

• **Bitcoin OTC** is a user-to-user trust network of Bitcoin users. The network is made bipartite by splitting each user into a 'rater' with all its outgoing edges and 'product' with all incoming edges. The ground truth is defined as: trustworthy users are the platform's founder and users he rated highly positively (≥ 0.5). Untrustworthy users are the ones that these trusted users give at least three more high negative ratings (≤ -0.5) than high positive ratings (≥ 0.5).

• **Bitcoin Alpha** is another Bitcoin trust network and its ground truth is created similar to OTC, starting from the founder of this platform.

7.5.2 Baselines

We compare FairJudge with nine state-of-the-art unsupervised and supervised algorithms. The **unsupervised** algorithms are:

• *Bias and Deserve (BAD)* [52] assigns a bias score $bias(u)$ to each user u, which measures user's tendency to give high or low ratings. $1 - |bias(u)|$ is the prediction made by BAD.

• *Trustiness* [54] algorithm assigns a trustiness, honesty and reliability score to each user, product and rating, respectively. We use the trustiness score as its prediction.

• *FraudEagle* [51] is a belief propagation based algorithm. Users are ranked according to their fraud score.

• *SpEagle* [53] incorporates behavior features into FraudEagle, and the final spam scores of users are used for ranking.

• *BIRDNEST* [48] ranks users by creating a Bayesian model with users' timestamp and rating distributions.

We also compare with **supervised** algorithms, when training labels are available:

• *SpEagle+* [53] is a supervised extension of SpEagle that leverages known training labels in the

ranking.

• *SpamBehavior* [44]: This technique uses user's average rating deviations as feature.

• *Spamicity* [56] is creates each user's features as its review burstiness and maximum reviews per day.

• *ICWSM'13* [55] uses user's fraction of positive reviews, maximum reviews in a day and average rating deviation as features for prediction.

7.5.3 Experiment 1: Unsupervised Prediction

In this experiment, the task is to rank the users based on their suspiciousness. We compare *unsupervised* FairJudge with the suite of five unsupervised algorithms in terms of their *Average Precision scores*, which measures the relative ordering the algorithm give to the known fair and unfair users. This score corresponds to the area under the precision-recall curve. We calculate two average precision scores – one for fair users and another for unfair users. Table 7.3 shows the resulting average precision score on the five datasets.

We see that FairJudge performs the best in identifying fair users in 4 out of 5 networks, and second best in the Flipkart network. In identifying unfair users, our algorithm performs the best in two networks and second best in the two Bitcoin networks. The BIRDNEST algorithm is observed to perform quite well in case of Flipkart network, but has much weaker performance on other datasets. Note that FraudEagle, SpEagle and Trustiness are not scalable and do not converge for the two largest networks, Epinions and Flipkart, as opposed to FairJudge which is guaranteed to converge. We discuss scalability in Section 7.5.7.

Overall, in unsupervised prediction, FairJudge performs the best or second best in 9 out of 10 cases.

	OTC	Alpha	Amazon	Epinions	Flipkart
FraudEagle	0.89	0.76	0.81	*nc*	*nc*
BAD	0.79	0.68	0.80	0.81	0.64
SpEagle	0.69	0.57	0.63	*nc*	*nc*
BIRDNEST	0.71	0.73	0.56	0.84	0.80
Trustiness	0.82	0.75	0.72	*nc*	*nc*
SpEagle+	0.55	0.66	0.67	*nc*	*nc*
SpamBehavior	0.77	0.69	0.80	0.80	0.60
Spamicity	0.88	0.74	0.60	0.50	0.82
ICWSM'13	0.75	0.71	0.84	0.82	0.82
FairJudge	0.91	0.85	0.86	0.89	0.85

Table 7.4: Supervised Predictions: 10-fold cross validation with individual predictions as features in a Random Forest classifier. Values reported are AUC. FairJudge performs the best across all datasets. *nc* means 'no convergence'.

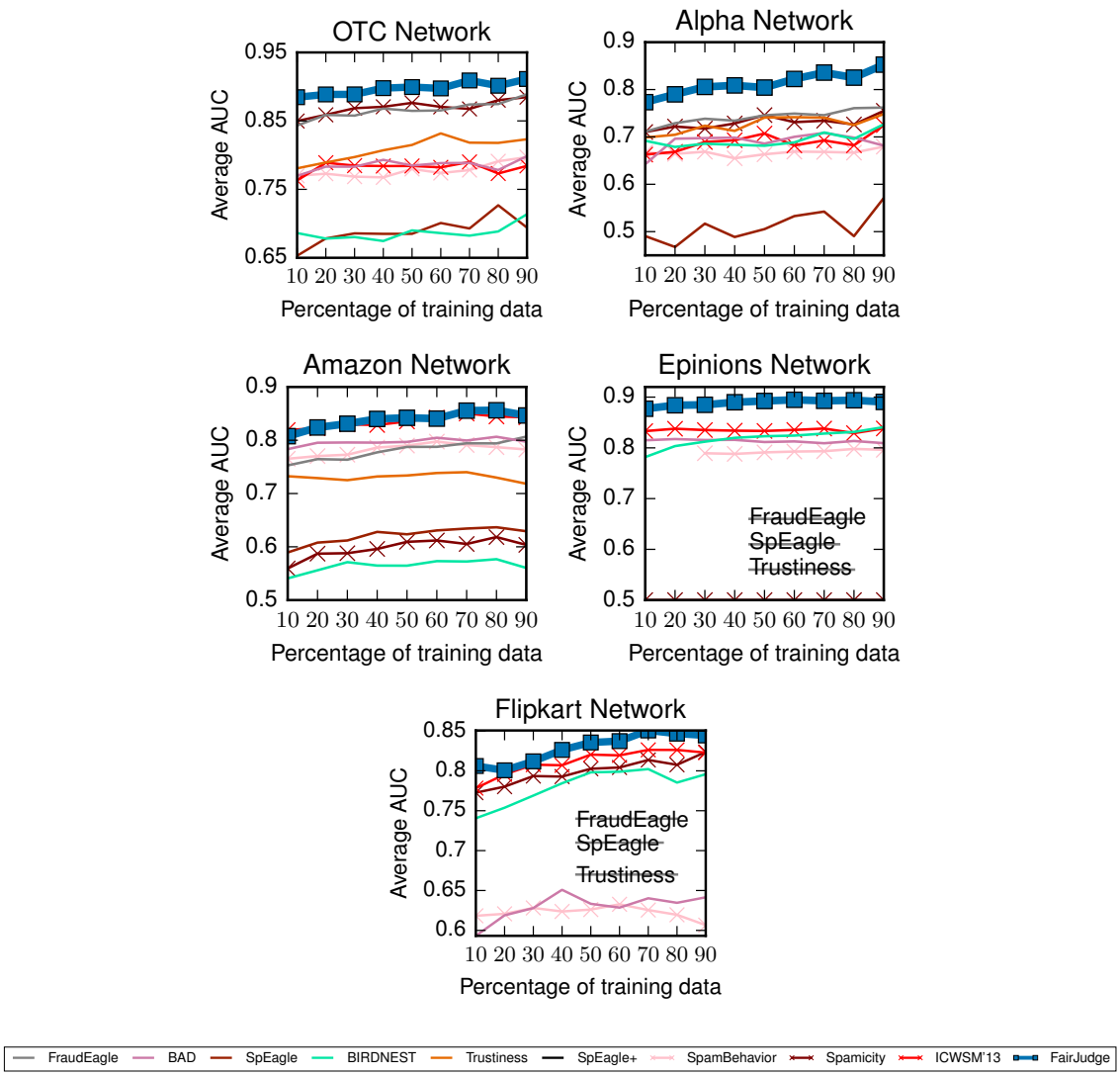


Figure 7.5: Variation of AUC with percentage of training data available for supervision. FairJudge consistently performs the best across all settings, and its performance is robust to the training percentage.

7.5.4 Experiment 2: Supervised Prediction

In this experiment, the task is to predict the malicious and benign users, given some labels from both categories. The performance is measured using *area under the ROC curve (AUC)* which is a standard measures when data is imbalanced, as is our case. For each algorithm, a feature vec-

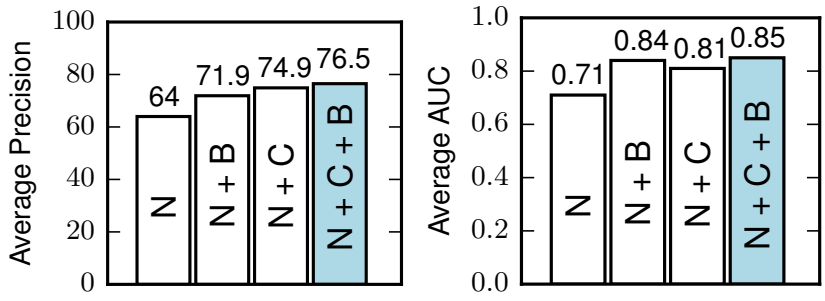
tor is created for each user and a binary classifier is trained. As a reminder from Section 7.4, for each user u, supervised FairJudge creates a 1296 dimentional feature vector of its fairness scores $F(u|\alpha_1, \alpha_2, \beta_1, \beta_2)$, one for each of the 1296 combinations of $\alpha_1, \alpha_2, \beta_1, \beta_2 \in [0, 5]$. For baselines FraudEagle, BAD, SpEagle, SpEagle+, BIRDNEST and Trustiness, the feature vector for user u is the score the baseline gives to u and u's outdegree. For baselines SpamBehavior, Spamicity and ICWSM'13, the feature vector for user u is their respective features given in Section 7.5.2.

We perform stratified 10-fold cross-validation using random forest classifier. The resulting AUCs are reported in Table 7.4. We see that FairJudge outperforms all existing algorithms across all datasets and consistently has AUC ≥ 0.85.

Interestingly, supervised FairJudge performs extremely well on the Flipkart dataset, while it did not perform as well on this dataset in the unsupervised experiment. This is because by using the training data, the classifier learns the importance of features $F(u|\alpha_1, \alpha_2, \beta_1, \beta_2)\ \forall\{\alpha_1, \alpha_2, \beta_1, \beta_2\} \in \mathcal{C}$. We reported the 100 most unfair users predicted by FairJudge to review fraud investigators in Flipkart, and they found 80 users to be fraudulent (80% accuracy).

7.5.5 Experiment 3: Robustness of FairJudge

In this experiment, we evaluate the performance of the algorithms as the percentage of training data changes. We vary the training data from 10% to 90% in steps of 10. Figure 7.5 shows the average AUC on test sets by using 50 random samples of training data. We make two observations. First, FairJudge is robust to the amount of training data. Its performance is relatively stable (AUC ≥ 0.80 in almost all cases) as the amount of training data varies. Second, FairJudge outperforms other algorithms consistently across all datasets for almost all training percentages. Together, these two show the efficiency of supervised FairJudge algorithm even



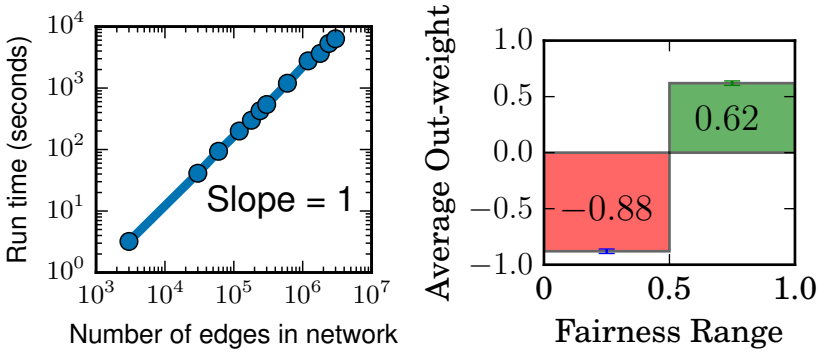
(a) Unsupervised (b) Supervised

Figure 7.6: Change in performance of FairJudge on Alpha network in (a) unsupervised and (b) supervised experiments when different components are used: network (N), cold start treatment (C) and behavioral (B).

when small amount of training data is available.

7.5.6 Experiment 4: Importance of Network, Cold Start and Behavior

In this experiment, we show the importance of the different components in the FairJudge algorithm – network (given by Equations 7.4, 7.5 and 7.6; shown as N), cold start treatment (C), and behavioral properties (B). Figure 7.6(a) shows the average precision in unsupervised case for the Alpha dataset, when network property is combined with the other two components, and Figure 7.6(b) shows the average AUC in the supervised case. In both cases, network properties alone has the lowest performance. Adding either the cold start treatment component or the behavioral property component increases this performance. To combine network properties with behavioral property component only, the cold start property is removed by setting $\alpha_1 = \beta_1 = 0$ in the FairJudge algorithm. Likewise, α_2 and β_2 are set to 0 to combine with cold start treatment alone. Further, combining all three together gives the best performance. Similar observations hold for the other datasets as well. This shows that all three components are important for predicting fraudulent users.



(a) (b)

Figure 7.7: (a) FairJudge scales linearly - the running time increases linearly with the number of edges. (b) Unfair users give highly negative ratings.

7.5.7 Experiment 5: Linear scalability

We have theoretically proved in Section 7.4.1 that FairJudge is linear in running time in the number of edges. To show this experimentally as well, we create random networks of increasing number of nodes and edges and compute the running time of the algorithm till convergence. Figure 7.7 shows that the running time increases linearly with the number of edges in the network, which shows that FairJudge algorithm is indeed scalable to large size networks for practical use.

7.5.8 Discoveries

Here we look at some insights and discoveries about malicious behavior found by FairJudge.

As seen previously in Figure 7.2, most ratings given by unfair users have low reliability, while some have high reliability, indicating camouflage to masquerade as fair users. At the same time, most ratings of fair users have high reliability, but some ratings have low reliability, indicating personal opinion about products that disagrees with 'consensus'.

We see in Figure 7.7(b) that in the Amazon network, unfair users detected by FairJudge

($F(u) \leq 0.5$) give highly negative rating on average (mean out-rating weight of -0.88), while fair users give positive ratings (mean out-weight 0.62). Unfair users also give almost half as many ratings as fair users (7.85 vs 15.92). This means on average, fraudsters aggressively bad-mouth target products.

We also observe in Figure 7.8 that unfair users in OTC network: (a) give ratings in quick succession of less than a few minutes, and (b) exhibit bimodal rating pattern – either they give all -1.0 ratings (possibly, bad-mouthing a competitor) or all +1.0 ratings (possibly, over-selling their products/friends). As an example, the most unfair user found by FairJudge had 3500 ratings, all with +0.5 score, and almost all given 15 seconds apart (apparently, a script). On the other hand, fair users have a day to a month between consecutive ratings, and they give mildly positive ratings (between 0.1 and 0.5). These observations are coherent with existing research [47, 78].

Figure 7.1(d) shows a set of 40 unfair users, as identified by FairJudge on the Alpha network, that collude to positively rate each other and form sets of tightly knit clusters – they are confirmed to be shills of a single user.

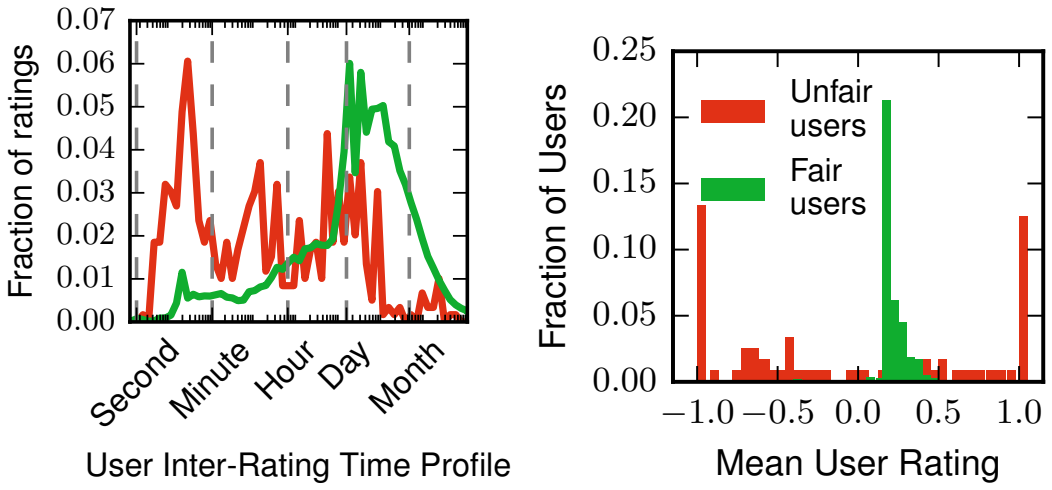
In summary, unfair users detected by FairJudge exhibit strange characteristics with respect to their behavior:

• They have bimodal rating pattern - they give too low (bad-mouthing) or too high (over-selling) ratings.

• They are less active, have no daily periodicity, and post quickly, often less than a few minutes apart.

• They tend to form near-cliques, colluding to boost their own or their product's ratings.

• They camouflage their behavior to masquerade as benign users.



(a) (b)

Figure 7.8: Identified unfair users by FairJudge are (a) faster in rating, and (b) give extreme ratings.

7.6 Conclusions

We presented the FairJudge algorithm to address the problem of identifying fraudulent users in rating networks. This chapter has the following contributions:

• **Algorithm**: We proposed three mutually-recursive metrics - fairness of users, goodness of products and reliability of ratings. We extended the metrics to incorporate Bayesian solutions to cold start problem and behavioral properties. We proposed the FairJudge algorithm to iteratively compute these metrics.

• **Theoretical guarantees**: We proved that FairJudge algorithm has linear time complexity and is guaranteed to converge in a bounded number of iterations.

• **Effectiveness**: By conducting five experiments, we showed that FairJudge outperforms nine existing algorithms to predict fair and unfair users. FairJudge is practically useful, and already under deployment at Flipkart.

Part IV

Concluding Remarks

Chapter 8

Conclusion, Impact and Future Directions

8.1 Conclusion

In this thesis, we studied various different types of malicious behavior on the web which manifests in the form of malicious users, such as trolls, vandals and sockpuppets, and as false information, such as hoaxes and fake reviews. The impact of online malicious behavior on people's lives have been detrimental, causing distress, offline harassment and even fatalities. However, there is a wide variety of malicious users and misinformation on the web, which are all fundamentally different from one another in terms of their objectives, targets, and operation. Overcoming the challenges of data imbalance, limitations of ground-truth, and deceptive behavior of malicious entities, we study malicious behavior across a wide variety of platforms namely Wikipedia, Slashdot, online discussion forums like CNN, IGN, Fox News, etc., and e-commerce and rating platforms like Epinions and Amazon, and ultimately develop models to detect them accurately, in order to maintain the safety, usability and quality of the web.

The underlying guiding principle behind the thesis is summarized as:

Malicious behavior differs from benign behavior both with respect to their attributes and connectivity. These differences can be used to create efficient computational models to detect them.

Adopting a data-driven approach, we characterize malicious behavior of trolls, vandals, sockpuppets, hoaxes and fake reviews in terms of their activities, linguistic, temporal, network and community feedback properties. Finally, we develop a suite of five algorithms that lie in two categories – feature-based and graph-based – for detecting malicious behavior across a wide variety of platforms.

In Chapter 3, we looked at vandals on Wikipedia, i.e., editors who make unconstructive edits to Wikipedia articles. By studying over 33,000 editors, we found striking differences between vandals and benign editors in their behavior in terms of the articles she edits, the relation between these articles, the burstiness of their edits, and the feedback by other editors. For instance, we found that vandals are faster in editing and make incoherent edits. By modeling several of these behavioral attributes as features, we developed a supervised machine learning model, called VEWS, which outperforms current vandalism detection methods, and takes *2.13 edits* on average to predict vandals.

Next, in Chapter 4, we studied over 21,000 hoax articles created on Wikipedia. We first found that a small fraction of hoaxes are highly impactful, as they survived for a long time, were viewed frequently, and were well cited from across the web. We studied the article structure and content, embeddedness in article network, and the editor's properties for the hoax and non-hoax articles, and found that hoaxes are created very differently from non-hoaxes. We created a highly accurate supervised machine learning model to identify hoax articles (*98% AUC*).

Further, in Chapter 5, we analysed sockpuppetry across nine popular online discussion fo-

rums. We found, for the first time, that sockpuppets may be created for both benign purposes, such as participating in different topics, and also for malicious purposes, mainly for showing excessive support for their opinions in arguments. Studying the writing style of sockpuppets, we found that they write worse (shorter sentences, more abusive, etc.) than ordinary users. We developed a taxonomy of sockpuppetry in two dimensions – (i) deceptiveness, i.e., whether they pretend to be different users, and (ii) supportiveness, i.e., if they support arguments of other sockpuppets controlled by the same user. Finally, we created a supervised model to identify whether a pair of accounts belongs to the same user with very high accuracy (*91% AUC*).

In Chapter 6, to identify trolls, we developed the decluttering algorithm that iteratively identifies and removes suspicious edges from the network, that trolls use to masquerade as benign users. Decluttering outperformed existing techniques with over *3 times the accuracy*, and it was *25-50 times faster* as well.

Lastly, in Chapter 7, we developed an algorithm called FairJudge to identify fraudulent reviewers in rating platforms. It is an iterative algorithm that uses the rating graph to calculate the fairness of each reviewer, the reliability of each rating and the quality of each product, simultaneously. We extended FairJudge to incorporate penalties for suspicious behavioral properties of reviewers and products. Overall, this algorithm outperformed several existing algorithms for predicting fake reviewers across many rating platforms, with over *85% AUC*.

Altogether, this thesis establishes comprehensive understanding of malicious behavior on the web. We develop a suite of five models and algorithms to accurately detect several distinct types of malicious behavior across various web platforms. The analysis leading to the algorithms develops interpretable understanding of malicious behavior on the web.

8.2 Impact

In addition to the intellectual contributions, this work has already had broad impact across academia and industry:

- **Academic Recognition**
  - Paper on sockpuppets, from Chapter 5, has won the Best Paper Honorable Mention Award at WWW 2017 conference.
  - Paper on vandals, from Chapter 3, has been replicated in parts in course projects at Birla Institute of Technology and Science (BITS) Pilani, India [27] and DJ Sanghvi College of Engineering, India [28].
  - Paper on Wikipedia hoaxes, from Chapter 4, has been included in the library guide of Hillsborough College [31].
- **Industry Adoption**
  - FairJudge algorithm, from Chapter 7, is being used in production at Flipkart, India's largest e-commerce platform.
  - VEWS algorithm, from Chapter 3, will be implemented in parts at the Wikimedia Foundation to support the English Wikipedia.
- **Media Recognition**
  - Research on sockpuppets, from Chapter 5, has been covered in New Scientist, TechCrunch and WOWScience magazines.
- **Education and Talks**

– Several papers of the thesis (Chapters 3–7) have been taught in tutorials at ASONAM 2016, CIKM 2016, and WWW 2017 conferences.

– Several papers of the thesis (Chapters 3–7) have been taught in undergraduate courses, graduate courses and seminars at University of Maryland's CMSC 422, 2017 Big Data winter school, Bari, Italy, University of Freiburg's Web Science Seminar, University of Waterloo's CS 698 and Social and Economic Networks, and University of Alberta's CMPUT 605 courses.

– Papers on Wikipedia vandals (Chapter 3) and hoaxes (Chapter 4) have been invited for talk at CyberSafety workshop at CIKM 2016, Wikipedia Research 2016, and VOGIN-IP conference 2017.

8.3 Future Directions

There are several important research directions that will further improve the understanding of malicious behavior and development of more accurate detection systems. In this thesis, we focused on several different types of malicious behavior, but in isolation and mostly assuming that malicious users work independently. These give rise to two important research problems: understanding malicious behavior across platforms and understanding the behavior of group and coordinated malicious behavior. Further, while most of the research work has aimed to identify malicious behavior empirically, there are two important aspects of malicious behavior that are understudied: adversarial malicious behavior and unified theory for maliciousness. We will take a brief look at each of these different directions below.

8.3.1 Malicious behavior across multiple platforms

Most research has focused on understanding and identifying malicious behavior on a single platform. Since it is trivial to create accounts in more than one platform, malicious users use multiple platforms to increase their effect. For example, people who write Wikipedia articles in exchange for money offer their services on more than one platform to increase their revenue, or spammers may post the same false information about certain product or services on Twitter, Reddit, and several discussion platforms. In our research, we had a brief glimpse into malicious behavior across platforms. We found that sockpuppets may exist beyond a single discussion community – for example, we found 14 different sockpuppet groups that were used in more than one online community. While past research has focused on identifying the same user with accounts on different platforms [136, 168–170], the detection of such accounts for malicious users may be strikingly different, as they may actively try to hide their identity.

Detecting malicious users across different platforms is an important problem due to its advantages. Such a study may allow us to better characterize how malicious behavior may change in different platforms. Further, understanding whether and how the same user changes his/her behavior to prevent detection by the platform-specific abuse prevention systems would give insights on active behavior of malicious users. This may lead to creation of platform-independent attributes that can be used to identify malicious behavior.

8.3.2 Group and coordinated malicious behavior

In this thesis, we focused on studying malicious behavior by mostly assuming that separate malicious users do not coordinate with each other. In more complicated scenarios, malicious users may coordinate in order to be more effective in reaching their objective. For example, in Chapter 7, we found that malicious users in Bitcoin trust networks rate each other positively, and

therefore increase their reputation. Such users may be the same person using multiple accounts and acting as a group, or it may be a group of separate people coordinating with each other. Research on identifying and separating the two cases would help in developing a clear understanding of the similarities and differences between the two cases, and their purpose and functionality. For instance, it may happen that one of the two cases is more frequently used by malicious users (potentially the former due to its simplicity), while the other is more benign. Doing real-world case studies on both of these behavior would help in understanding the underground market behind malicious behavior (e.g., buying fake followers on Twitter). While we focused on the former by studying sockpuppets in Chapter 5, our analysis was mostly limited to pair of accounts being used by the same person, and it was done only on online discussion platforms. Understanding how larger groups of sockpuppets function may reveal additional ways in which sockpuppets may coordinate, and may allow us to observe more pronounced effects of sockpuppetry.

8.3.3 Adversarial model of maliciousness

Malicious users can actively change their behavior in order to avoid detection. However, most research assumes a fixed behavior of malicious users, which comes as an artifact of using historical data. Any detection algorithm that identifies malicious behavcior using the data, even if it works well, may not be effective when implemented as malicious users may change their behavior and evade detection. Such actively adaptive behavior of maliciousness has not been studied in great detail.

In this direction, theoretical as well as empirical studies on how malicious behavior changes when subjected to a detection system would be very useful and practical. This is an iterative process – a change in malicious behavior would require a change in the detection system, which would further trigger a change in behavior of malicious users. This indicates towards a game-

theoretic model between malicious users and the detection system.

8.3.4 Unified theory of maliciousness

Currently, malicious behavior in one platform is considered to be distinct from that on another platform. For instance, vandals on Wikipedia make unconstructive edits in Wikipedia, and are quite different from trolls, who harass other users, or from reviewers who give fake reviews to certain products on e-commerce websites. While they may be different from each other in several respects, some commonalities have been identified independently. For example, fake reviewers, vandals and hoaxsters have all been found to be faster in their activity than benign users. This suggests there may be a unifying objective that several types of malicious users follow.

A comprehensive study between the similarities and differences between several type of malicious behavior would lead to great insight into this. This could lead to generation of a general model that could be used to detect multiple type of maliciousness, irrespective of the platform. Platform-specific features may lead to further increase in the detection accuracy. Therefore, a unified theory that encompasses several different types of malicious behavior would be very useful to develop.

Part V

Appendices

Appendix 9:

Proofs for the FairJudge Algorithm

Let us first set up the basics that we need in order to prove the lemma and the two theorems in Chapter 7. As a reminder, $F^t(u), G^t(p)$ and $R^t(u,p)$ represent the fairness score of user u, goodness score of product p and reliability score of rating r at the end of iteration t of Algorithm 1 (Chapter 7), for some input $\alpha_1, \alpha_2, \beta_1, \beta_2$. Recall that $\alpha_1, \alpha_2, \beta_1$ and β_2 are non-negative integers. $F^\infty(u), G^\infty(p)$ and $R^\infty(u,p)$ are their final values after convergence for the same $\alpha_1, \alpha_2, \beta_1, \beta_2$.

At the end of iteration t of Algorithm 1, and by Equations 7.1, 7.2, and 7.3 in Figure 7.4, we get,

$$F^t(u) = \frac{0.5 \cdot \alpha_1 + \alpha_2 \cdot IBIRDNEST_{IRTD_U}(u) + \sum_{(u,p') \in \texttt{Out(u)}} R^t(u,p')}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|} \quad (9.1)$$

$$R^t(u,p) = \frac{1}{2}(F^{t-1}(u) + (1 - \frac{|\texttt{score(u,p)} - G^t(p)|}{2})) \quad (9.2)$$

$$G^t(p) = \frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum_{(u',p) \in \texttt{In(p)}} R^{t-1}(u',p) \cdot \texttt{score(u',p)}}{\beta_1 + \beta_2 + |\texttt{In(p)}|} \quad (9.3)$$

When the algorithm converges,

$$F^{\infty}(u) = \frac{0.5 \cdot \alpha_1 + \alpha_2 \cdot IBIRDNEST_{IRTD_U}(u) + \sum_{(u,p') \in \texttt{Out(u)}} R^{\infty}(u,p')}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|} \quad (9.4)$$

$$R^{\infty}(u,p) = \frac{1}{2}(F^{\infty}(u) + (1 - \frac{|\texttt{score(u,p)} - G^{\infty}(p)|}{2})) \quad (9.5)$$

$$G^{\infty}(p) = \frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum_{(u',p) \in \texttt{In(p)}} R^{\infty}(u',p) \cdot \texttt{score(u',p)}}{\beta_1 + \beta_2 + |\texttt{In(p)}|} \quad (9.6)$$

9.1 Proof for Lemma 1: Lemma 1

Lemma 2 (Lemma 1) *The difference between a product p's final goodness score and its score after the first iteration is at most 1, i.e.* $|G^{\infty}(p) - G^{1}(p)| \leq 1$. *Similarly,* $|R^{\infty}(u,p) - R^{1}(u,p)| \leq 3/4$ *and* $|F^{\infty}(u) - F^{t}(u)| \leq 3/4$.

Proof.

During initialization, all the ratings have reliability scores of 1, i.e. $R^{0}(u,p) = 1, \forall (u,p) \in \mathcal{R}$, and this is used to calculate the value of G^{1}.

Now, we will prove that $|G^{\infty}(p) - G^{1}(p)| \leq 1$.

From Equations 9.3 and 9.6, and substituting $t = 1$, we get,

$$|G^{\infty}(p) - G^{1}(p)| = |\frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum\limits_{(u',p) \in \texttt{In(p)}} R^{\infty}(u',p) \cdot \texttt{score(u',p)}}{\alpha_2 + \beta_2 + |\texttt{In(p)}|}$$

$$- \frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum\limits_{(u',p) \in \texttt{In(p)}} R^{0}(u',p) \cdot \texttt{score(u',p)}}{\alpha_2 + \beta_2 + |\texttt{In(p)}|}|$$

$$\Rightarrow |G^{\infty}(p) - G^{1}(p)| = |\frac{\sum\limits_{(u',p) \in \texttt{In(p)}} (R^{\infty}(u',p) - R^{0}(u',p)) \cdot \texttt{score(u',p)}}{\alpha_2 + \beta_2 + |\texttt{In(p)}|}|$$

Since $|x + y| \leq |x| + |y|$, we have,

$$|G^{\infty}(p) - G^{1}(p)| \leq \frac{\sum\limits_{(u',p) \in \texttt{In(p)}} |(R^{\infty}(u',p) - R^{0}(u',p)) \cdot \texttt{score(u',p)}|}{\alpha_2 + \beta_2 + |\texttt{In(p)}|}$$

Since $|x \cdot y| = |x| \cdot |y|$, we get,

$$|G^{\infty}(p) - G^{1}(p)| \leq \frac{\sum\limits_{(u',p) \in \texttt{In(p)}} |(R^{\infty}(u',p) - R^{0}(u',p))| \cdot |\texttt{score(u',p)}|}{\alpha_2 + \beta_2 + |\texttt{In(p)}|}$$

Now $|(R^{\infty}(u',p) - R^{0}(u',p))| \leq 1$, as all reliability scores lie in [0,1], and $|\texttt{score(u',p)}| < 1$.

So we get,

$$|G^{\infty}(p) - G^{1}(p)| \leq \frac{\sum\limits_{(u',p) \in \texttt{In(p)}} 1}{\alpha_2 + \beta_2 + |\texttt{In(p)}|} = \frac{|\texttt{In(p)}|}{\alpha_2 + \beta_2 + |\texttt{In(p)}|} \leq 1$$

.

$$\Rightarrow |G^{\infty}(p) - G^{1}(p)| \leq 1$$

■

9.2 Proof for Theorem 1: Convergence Theorem

Theorem 2 (Convergence Theorem) *The difference during iterations is bounded as* $|R^{\infty}(u,p) - R^{t}(u,p)| \leq (\frac{3}{4})^{t}, \forall (u,p) \in \mathcal{R}$. *As* t *increases, the difference decreases and* $R^{t}(u,p)$ *converges to* $R^{\infty}(u,p)$. *Similarly,* $|F^{\infty}(u) - F^{t}(u)| \leq (\frac{3}{4})^{t}, \forall u \in \mathcal{U}$ *and* $|G^{\infty}(p) - G^{t}(p)| \leq (\frac{3}{4})^{(t-1)}, \forall p \in \mathcal{P}$.

Proof. We will prove the convergence of the rating reliability values using mathematical induction.

From Equations 9.2 and 9.5,

$$|R^{\infty}(u,p) - R^{t}(u,p)| = |\tfrac{1}{2}(F^{\infty}(u) + (1 - \tfrac{|\texttt{score(u,p)} - G^{\infty}(p)|}{2}))$$

$$-\tfrac{1}{2}(F^{t-1}(u) + (1 - \tfrac{|\texttt{score(u,p)} - G^{t}(p)|}{2}))|$$

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| = \tfrac{1}{2} \cdot |(F^{\infty}(u) - F^{t-1}(u)) + (\tfrac{|\texttt{score(u,p)} - G^{t}(p)| - |\texttt{score(u,p)} - G^{\infty}(p)|}{2})|$$

As $|x| - |y| \leq |x - y|$, applying it to the goodness terms,

$$|R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{1}{2} \cdot |(F^{\infty}(u) - F^{t-1}(u)) + \frac{|G^{\infty}(p) - G^{t}(p)|}{2}|$$

Replacing $F^{\infty}(u)$ and $F^{t-1}(u)$ using Equations 9.1 and 9.4

$$|R^{\infty}(u,p) - R^{t}(u,p)| \leq \tfrac{1}{2} \cdot (|\frac{0.5 \cdot \alpha_1 + \alpha_2 \cdot IBIRDNEST_{IRTD_U}(u) + \sum\limits_{r' \in \texttt{Out(u)}} R^{\infty}(r')}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|}$$

$$- \frac{0.5 \cdot \alpha_1 + \alpha_2 \cdot IBIRDNEST_{IRTD_U}(u) + \sum\limits_{r' \in \texttt{Out(u)}} R^{t-1}(r'))}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|} + \tfrac{|G^{\infty}(p) - G^{t}(p)|}{2}|)$$

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \tfrac{1}{2} \cdot (|\frac{\sum\limits_{r' \in \texttt{Out(u)}} (R^{\infty}(r') - R^{t-1}(r'))}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|} + \tfrac{|G^{\infty}(p) - G^{t}(p)|}{2}|)$$

As $|a + b| \leq |a| + |b|$,

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \tfrac{1}{2} \cdot (|\frac{\sum\limits_{r' \in \texttt{Out(u)}} (R^{\infty}(r') - R^{t-1}(r'))}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|}| + |\tfrac{|G^{\infty}(p) - G^{t}(p)|}{2}|)$$

Again, as $|x + y| \leq |x| + |y|$,

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \tfrac{1}{2} \cdot (\frac{\sum\limits_{r' \in \texttt{Out(u)}} |R^{\infty}(r') - R^{t-1}(r')|}{|\alpha_1 + \alpha_2 + |\texttt{Out(u)}||} + \tfrac{|G^{\infty}(p) - G^{t}(p)|}{2}) \qquad (9.7)$$

Base case of induction.

We need to prove in the base case that, $|R^{\infty}(u,p) - R^{1}(u,p)| < 3/4$.

Substituting t = 1 in Equation 9.7,

$$|R^{\infty}(u,p) - R^{1}(u,p)| \leq \frac{1}{2} \cdot (\frac{\sum\limits_{r' \in \texttt{Out(u)}} |R^{\infty}(r') - R^{0}(r')|}{|\alpha_1 + \alpha_2 + |\texttt{Out(u)}||} + \frac{|G^{\infty}(p) - G^{1}(p)|}{2})$$

Trivially, $|R^{\infty}(r') - R^{0}(r')| < 1$, since reliability scores always lie in [0,1]. And from Lemma 1,

$|G^{\infty}(p) - G^{1}(p)| < 1$. So,

$$|R^{\infty}(u,p) - R^{1}(u,p)| \leq \frac{1}{2} \cdot (\frac{\sum\limits_{r' \in \texttt{Out(u)}} 1}{|\alpha_1+\alpha_2+|\texttt{Out(u)}||} + \frac{1}{2})$$

$$\Rightarrow |R^{\infty}(u,p) - R^{1}(u,p)| \leq \frac{1}{2} \cdot (\frac{|\texttt{Out(u)}|}{|\alpha_1+\alpha_2+|\texttt{Out(u)}||} + \frac{1}{2})$$

$$\Rightarrow |R^{\infty}(u,p) - R^{1}(u,p)| \leq \frac{1}{2} \cdot (1 + \frac{1}{2})$$

$$\Rightarrow |R^{\infty}(u,p) - R^{1}(u,p)| \leq \frac{3}{4}$$

Induction Step.

We assume in the induction step that $|R^{\infty}(u,p) - R^{(t-1)}(u,p)| \leq \frac{3}{4}^{(t-1)}, \forall (u,p) \in \mathcal{R}$, which is consistent with the base case already.

We have to prove, $|R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{3}{4}^{(t)}$

We know from Equation 9.7, that

$$|R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{1}{2} \cdot (\frac{\sum\limits_{r' \in \texttt{Out(u)}} |R^{\infty}(r') - R^{t-1}(r')|}{\alpha_1 + \alpha_2 + |\texttt{Out(u)}|} + \frac{|G^{\infty}(p) - G^{t}(p)|}{2})$$

We first find the bounds for $|G^{\infty}(p) - G^{t}(p)|$. From Equations 9.3 and 9.6, we get,

$$|G^{\infty}(p) - G^{t}(p)| = |\frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum_{(u',p) \in \texttt{In(p)}} R^{\infty}(u',p) \cdot \texttt{score(u',p)}}{\beta_1+\beta_2+|\texttt{In(p)}|}$$

$$- \frac{\beta_2 \cdot IBIRDNEST_{IRTD_P}(p) + \sum_{(u',p) \in \texttt{In(p)}} R^{t-1}(u',p) \cdot \texttt{score(u',p)}}{\beta_1+\beta_2+|\texttt{In(p)}|}|$$

$$\Rightarrow |G^{\infty}(p) - G^{t}(p)| = |\frac{\sum_{(u',p) \in \texttt{In(p)}} (R^{\infty}(u',p) - R^{t-1}(u',p)) \cdot \texttt{score(u',p)}}{\beta_1+\beta_2+|\texttt{In(p)}|}|$$

As $|x + y| \leq |x| + |y|$,

$$\Rightarrow |G^{\infty}(p) - G^{t}(p)| \leq \frac{\sum_{(u',p)\in \texttt{In(p)}} |(R^{\infty}(u',p) - R^{t-1}(u',p)) \cdot \texttt{score(u',p)}|}{|\beta_1 + \beta_2 + |\texttt{In(p)}||}$$

As $|x.y| = |x|.|y|$,

$$\Rightarrow |G^{\infty}(p) - G^{t}(p)| \leq \frac{\sum_{(u',p)\in \texttt{In(p)}} |(R^{\infty}(u',p) - R^{t-1}(u',p))| \cdot |\texttt{score(u',p)}|}{|\beta_1 + \beta_2 + |\texttt{In(p)}||}$$

As $|\texttt{score(u',p)}| \leq 1, \forall (u',p) \in \mathcal{R}$ and $|(R^{\infty}(u',p) - R^{t-1}(u',p))| \leq \frac{3}{4}^{t-1}$, by induction assumption:

$$\Rightarrow |G^{\infty}(p) - G^{t}(p)| \leq \frac{\sum_{(u',p)\in \texttt{In(p)}} \frac{3}{4}^{t-1}}{|\beta_1 + \beta_2 + |\texttt{In(p)}||}$$

$$\Rightarrow |G^{\infty}(p) - G^{t}(p)| \leq \frac{3}{4}^{t-1} \cdot \frac{\sum_{(u',p)\in \texttt{In(p)}} 1}{|\beta_1 + \beta_2 + |\texttt{In(p)}||}$$

$$\Rightarrow |G^{\infty}(p) - G^{t}(p)| \leq \frac{3}{4}^{t-1} \cdot \frac{|\texttt{In(p)}|}{|\beta_1 + \beta_2 + |\texttt{In(p)}||}$$

$$\Rightarrow |G^{\infty}(p) - G^{t}(p)| \leq \frac{3}{4}^{t-1}$$

Substituting this back into Equation 9.7, and since by induction assumption $|R^{\infty}(u,p) - R^{t-1}(u,p)| \leq \frac{3}{4}^{t-1}$, we get

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{1}{2} \cdot (\frac{\sum_{r' \in \texttt{Out(u)}} \frac{3}{4}^{t-1}}{|\alpha_1+\alpha_2+|\texttt{Out(u)}||} + \frac{1}{2} \cdot \frac{3}{4}^{t-1})$$

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{1}{2} \cdot (\frac{3}{4}^{t-1} \cdot \frac{\sum_{r' \in \texttt{Out(u)}} 1}{|\alpha_1+\alpha_2+|\texttt{Out(u)}||} + \frac{1}{2} \cdot \frac{3}{4}^{t-1})$$

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{1}{2} \cdot (\frac{3}{4}^{t-1} \cdot \frac{|\texttt{Out(u)}|}{|\alpha_1+\alpha_2+|\texttt{Out(u)}||} + \frac{1}{2} \cdot \frac{3}{4}^{t-1})$$

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{1}{2} \cdot (\frac{3}{4}^{t-1} + \frac{1}{2} \cdot \frac{3}{4}^{t-1}) = \frac{1}{2} \cdot (\frac{3}{2} \cdot \frac{3}{4}^{t-1})$$

$$\Rightarrow |R^{\infty}(u,p) - R^{t}(u,p)| \leq \frac{3}{4}^{t}$$

The proofs for $|F^{\infty}(u) - F^{t}(u)| \leq (\frac{3}{4})^{t}, \forall u \in \mathcal{C}$ follows from the above proof.

As t increases, $(\frac{3}{4})^{t} \to 0$ and $(\frac{3}{4})^{t-1} \to 0$, so after t iterations, $R^{t}(u,p) \to R^{\infty}(u,p), \forall (u,p) \in \mathcal{R}$. So after t iterations, the algorithm converges.

The entire of the above proof holds for any combination of parameters $(\alpha_1, \alpha_2, \beta_1, \beta_2) \in \mathcal{C}$, therefore the entire algorithm converges for some t iterations. ■

9.3 Proof for Corrolary 1: Iterations till Convergence

Corollary 2 (Iterations till Convergence) *The number of iterations needed to reach convergence is at most* $2 + \lceil \frac{\log(\epsilon/2)}{\log(3/4)} \rceil$. *In other words, treating* ϵ *as constant, the number of iterations needed to reach convergence is bounded by a constant.*

Proof. Let $t = \lceil \frac{\log(\epsilon/2)}{\log(3/4)} \rceil$. By Theorem 1, after $t+1$ iterations, $\forall p \in \mathcal{P}$, $|G^{\infty}(p) - G^{t+1}(p)| \leq$

$(\frac{3}{4})^t \leq (\frac{3}{4})^{\log_{3/4}(\epsilon/2)} = \epsilon/2$. Similarly $|G^{\infty}(p) - G^{t+2}(p)| \leq \epsilon/2 \cdot 3/4 \leq \epsilon/2$. Thus,

$$|G^{t+1}(p) - G^{t+2}(p)| = |G^{t+1}(p) - G^{\infty}(p) + G^{\infty}(p) - G^{t+2}(p)|$$

As $|x + y| \leq |x| + |y|$, we get

$$\Rightarrow |G^{t+1}(p) - G^{t+2}(p)| \leq |G^{t+1}(p) - G^{\infty}(p)| + |G^{\infty}(p) - G^{t+2}(p)|$$

$$\Rightarrow |G^{t+1}(p) - G^{t+2}(p)| \leq |G^{\infty}(p) - G^{t+1}(p)| + |G^{\infty}(p) - G^{t+2}(p)| \leq 2\epsilon/2 = \epsilon.$$

Similarly, for fairness, $|F^{t+1}(u) - F^{t+2}(u)| \leq \epsilon$ for all users u, and the same holds for reliability.

Thus, by the termination condition of Algorithm 1, convergence occurs after at most $t + 2$ iterations. ■

Bibliography

[1] Erik Qualman. *Socialnomics: How social media transforms the way we live and do business*. John Wiley & Sons, 2010.

[2] Huan Liu, Fred Morstatter, Jiliang Tang, and Reza Zafarani. The good, the bad, and the ugly: uncovering novel research opportunities in social media mining. *International Journal of Data Science and Analytics*, 1(3-4):137–143, 2016.

[3] Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. *Social media mining: an introduction*. Cambridge University Press, 2014.

[4] Alex Leavitt, Evan Burchard, David Fisher, and Sam Gilbert. The influentials: New approaches for analyzing influence on twitter. *Web Ecology Project*, 4(2):1–18, 2009.

[5] Social networking to the rescue. http://www.womansday.com/health-fitness/a1993/social-networking-to-the-rescue-111876/, 2010.

[6] How social media is saving lives. http://www.socialmediatoday.com/social-networks/how-social-media-saving-lives, 2016.

[7] Anne P Mintz. *Web of deception: Misinformation on the Internet*. Information Today, Inc., 2002.

[8] Charles Seife. *Virtual Unreality: Just Because the Internet Told You, how Do You Know It's True?* Penguin, 2014.

[9] Michail Tsikerdekis and Sherali Zeadally. Online deception in social media. *Communications of the ACM*, 57(9):72–80, 2014.

[10] How many of the internets users are fake. http://www.dailyinfographic.com/how-many-of-the-internets-users-are-fake, 2014.

[11] Katharina Krombholz, Dieter Merkl, and Edgar Weippl. Fake identities in social media: A case study on the sustainability of the facebook business model. *Journal of Service Science Research*, 4(2):175–212, 2012.

[12] Michael Luca and Georgios Zervas. Fake it till you make it: Reputation, competition, and yelp review fraud. *Management Science*, 62(12):3412–3427, 2016.

[13] Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. Vews: A wikipedia vandal early warning system. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[14] An old swindle revived.; the "spanish prisoner" and buried treasure bait again being offered to unwary americans. https://query.nytimes.com/gst/abstract.html?res=980CE5D71638E433A25753C2A9659C94699ED7CF, 1898.

[15] Judith S Donath et al. Identity and deception in the virtual community. *Communities in cyberspace*, 1999.

[16] Peter Steiner. On the internet, nobody knows youre a dog. *The New Yorker*, 69(20):61, 1993.

[17] John Suler. The online disinhibition effect. *Cyberpsychology & behavior*, 7(3):321–326, 2004.

[18] Online harassment, pew research center. http://www.pewinternet.org/2014/10/22/online-harassment, 2014.

[19] Yavuz Akbulut, Yusuf Levent Sahin, and Bahadir Eristi. Cyberbullying victimization among turkish online social utility members. *Educational Technology & Society*, 13(4):192–201, 2010.

[20] David Wiener. Negligent publication of statements posted on electronic bulletin boards: Is there any liability left after zeran. *Santa Clara L. Rev.*, 39:905, 1998.

[21] Sameer Hinduja and Justin W Patchin. Offline consequences of online victimization: School violence and delinquency. *Journal of school violence*, 6(3):89–112, 2007.

[22] Sameer Hinduja and Justin W Patchin. Bullying, cyberbullying, and suicide. *Archives of suicide research*, 14(3):206–221, 2010.

[23] Fbi internet crime complaint center (ic3) annual reports. https://www.ic3.gov/media/annualreports.aspx, 2017.

[24] Erin E Buckels, Paul D Trapnell, and Delroy L Paulhus. Trolls just want to have fun. *Personality and individual Differences*, 67:97–102, 2014.

[25] VS Subrahmanian and Srijan Kumar. Predicting human behavior: The next frontiers. *Science*, 355(6324):489–489, 2017.

[26] http://en.wikipedia.org/wiki/Wikipedia:Vandalism.

[27] Mohit Gupta. Detecting vandalism in wikipedia. *Birla Institute of Technology and Science (BITS), Pilani, India*, 2016. Course project, Data Mining course.

[28] Ronak Dedhia. Vandalism detection in wikipedia. *DJ Sanghvi College of Engineering, Mumbai, India*, 2016. Course Project, Machine Learning.

[29] Wikipedia. Do not create hoaxes. Website, 2015. https://en.wikipedia.org/w/index.php?title=Wikipedia:Do_not_create_hoaxes&oldid=684241383 (accessed Oct. 16, 2015).

[30] Srijan Kumar, Robert West, and Jure Leskovec. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[31] Hillsborough Community College Library. Fake news, misleading news, biased news: Evaluating sources. Website, 2017. http://libguides.hccfl.edu/fakenews/evaluatingsources.

[32] Sadia Afroz, Michael Brennan, and Rachel Greenstadt. Detecting hoaxes, frauds, and deception in writing style online. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 461–475, 2012.

[33] Brad Stone and Matt Richtel. The hand that controls the sock puppet could get slapped. *New York Times*, 2007.

[34] Thamar Solorio, Ragib Hasan, and Mainul Mizan. A case study of sockpuppet detection in wikipedia. In *Workshop on Language Analysis in Social Media*, 2013.

[35] Srijan Kumar, Justin Cheng, Jure Leskovec, and V.S. Subrahmanian. An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[36] Kahina Gani, Hakim Hacid, and Ryan Skraba. Towards multiple identity detection in social networks. In *Proceedings of the 21st International Conference on World Wide Web*, 2012.

[37] Michail Tsikerdekis and Sherali Zeadally. Multiple account identity deception detection in social media using nonverbal behavior. *IEEE Transactions on Information Forensics and Security*, 9(8):1311–1321, 2014.

[38] Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. Accurately detecting trolls in slashdot zoo via decluttering. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, 2014.

[39] Theodoros Lappas, Gaurav Sabnis, and Georgios Valkanas. The impact of fake reviews on online visibility: A vulnerability assessment of the hotel industry. *INFORMS*, 27(4), 2016.

[40] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM*, 2008.

[41] Jing Wang, Anindya Ghose, and Panos Ipeirotis. Bonus, disclosure, and choice: what motivates the creation of high-quality paid reviews? In *ICIS*, 2012.

[42] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[43] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE International Conference on*. IEEE, 2016.

[44] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *CIKM*, 2010.

[45] Amanda J Minnich, Nikan Chavoshi, Abdullah Mueen, Shuang Luan, and Michalis Faloutsos. Trueview: Harnessing the power of multiple review sites. In *WWW*, 2015.

[46] Bimal Viswanath, M Ahmad Bashir, Mark Crovella, Saikat Guha, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. Towards detecting anomalous user behavior in online social networks. In *USENIX Security*, 2014.

[47] Huayi Li, Geli Fei, Shuai Wang, Bing Liu, Weixiang Shao, Arjun Mukherjee, and Jidong Shao. Bimodal distribution and co-bursting in review spam detection. In *WWW*, 2017.

[48] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Gunneman, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. Birdnest: Bayesian inference for ratings-fraud detection. In *SDM*, 2016.

[49] Bimal Viswanath, Muhammad Ahmad Bashir, Muhammad Bilal Zafar, Simon Bouget, Saikat Guha, Krishna P Gummadi, Aniket Kate, and Alan Mislove. Strength in numbers: Robust tamper detection in crowd computations. In *COSN*, 2015.

[50] Guangyu Wu, Derek Greene, and Pádraig Cunningham. Merging multiple criteria to identify suspicious reviews. In *RecSys*, 2010.

[51] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. In *ICWSM*, 2013.

[52] Abhinav Mishra and Arnab Bhattacharya. Finding the bias and prestige of nodes in networks based on trust scores. In *WWW*, 2011.

[53] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *KDD*, 2015.

[54] Guan Wang, Sihong Xie, Bing Liu, and S Yu Philip. Review graph based online store review spammer detection. In *ICDM*, 2011.

[55] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. What yelp fake review filter might be doing? In *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[56] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Spotting opinion spammers using behavioral footprints. In *KDD*, 2013.

[57] Richard Gilbert, Vandana Thadani, Caitlyn Handy, Haley Andrews, Tristan Sguigna, Alex Sasso, and Stephanie Payne. The psychological functions of avatars and alt (s): A qualitative study. *Computers in Human Behavior*, 32:1–8, 2014.

[58] Richard L Gilbert, Jessica A Foss, and Nora A Murphy. Multiple personality order: Physical and personality characteristics of the self, primary avatar and alt. In *Reinventing ourselves: Contemporary concepts of identity in virtual worlds*, pages 213–234. Springer, 2011.

[59] Yasmin B Kafai, Deborah A Fields, and Melissa Cook. Your second selves: avatar designs and identity play in a teen virtual world. In *Proceedings of DIGRA*, volume 2007, 2007.

[60] Avner Caspi and Paul Gorsky. Online deception: Prevalence, motivation, and emotion. *CyberPsychology & Behavior*, 9(1):54–59, 2006.

[61] Matt Richtel Brad Stone. The hand that controls the sock puppet could get slapped. Website, 2007. http://www.nytimes.com/2007/07/16/technology/16blog.html?pagewanted=all.

[62] Grace M Duffield and Peter Nils Grabosky. *The psychology of fraud*, volume 199. Australian Institute of Criminology Canberra, 2001.

[63] Cecil Eng Huang Chua, Jonathan Wareham, and Daniel Robey. The role of online trading communities in managing internet auction fraud. *MIS Quarterly*, pages 759–781, 2007.

[64] Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of the 20th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 2017.

[65] Xinran Chen, Sei-Ching Joanna Sin, Yin-Leng Theng, and Chei Sian Lee. Why do social media users share misinformation? In *JCDL*, 2015.

[66] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. The spreading of misinformation online. *PNAS*, 113(3):554–559, 2016.

[67] Adrien Friggeri, Lada Adamic, Dean Eckles, and Justin Cheng. Rumor cascades. In *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[68] Aditi Gupta, Hemank Lamba, Ponnurangam Kumaraguru, and Anupam Joshi. Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. In *Proceedings of the 22nd international conference on World Wide Web*, 2013.

[69] Qiongkai Xu and Hai Zhao. Using deep linguistic features for finding deceptive opinion spam. In *COLING*, 2012.

[70] Floyd H Allport and Milton Lepkin. Wartime rumors of waste and special privilege: why some people believe them. *The Journal of Abnormal and Social Psychology*, 40(1):3, 1945.

[71] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Communications of the ACM*, 59(7):96–104, 2016.

[72] Lina Zhou and Yu-wei Sung. Cues to deception in online chinese groups. In *Hawaii international conference on system sciences*, 2008.

[73] Catalina L Toma and Jeffrey T Hancock. What lies beneath: The linguistic traces of deception in online dating profiles. *Journal of Communication*, 62(1):78–97, 2012.

[74] Michael Brennan, Sadia Afroz, and Rachel Greenstadt. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):12, 2012.

[75] Patrick Juola. Detecting stylistic deception. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, 2012.

[76] David B Buller and Judee K Burgoon. Interpersonal deception theory. *Communication theory*, 6(3):203–242, 1996.

[77] Debora Donato, Stefano Leonardi, and Mario Paniccia. Combining transitive trust and negative opinions for better reputation management in social networks. In *SNA KDD*, 2008.

[78] Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. Antisocial behavior in online discussion communities. In *Proceedings of the Ninth International AAAI Conference on Web and Social Media*, 2015.

[79] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V.S. Subrahmanian. Fairjudge: Trustworthy user prediction in rating platforms. 2017.

[80] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 273–274. International World Wide Web Conferences Steering Committee, 2016.

[81] VS Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. The darpa twitter bot challenge. *Computer*, 49(6):38–46, 2016.

[82] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. Review spam detection via temporal pattern discovery. In *KDD*, 2012.

[83] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st international conference on World Wide Web*, 2012.

[84] B Adler, Luca De Alfaro, and Ian Pye. Detecting wikipedia vandalism using wikitrust. *Notebook papers of CLEF*, 1:22–23, 2010.

[85] Jacob Ratkiewicz, Michael Conover, Mark R Meiss, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Detecting and tracking political abuse in social media. *ICWSM*, 11:297–304, 2011.

[86] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Catchsync: catching synchronized behavior in large directed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[87] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. Understanding and combating link farming in the twitter social network. In *WWW*, 2012.

[88] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, 2010.

[89] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Inferring lockstep behavior from connectivity pattern in large graphs. *Springer KAIS*, pages 1–30, 2015.

[90] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web*, 2013.

[91] Jennifer Ann Golbeck. *COMPUTING AND APPLYING TRUST IN WEB-BASED SOCIAL NETWORKS*. PhD thesis, University of Maryland, USA, 2005.

[92] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, pages 640–651, 2003.

[93] Ramanathan V. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *WWW*, pages 403–412, 2004.

[94] Zhaoming Wu, Charu C Aggarwal, and Jimeng Sun. The troll-trust model for ranking in signed networks. In *WSDM*, 2016.

[95] Rong-Hua Li, Jeffery Xu Yu, Xin Huang, and Hong Cheng. Robust reputation-based ranking on bipartite rating networks. In *SDM*, 2012.

[96] Cheng Chen, Kui Wu, Venkatesh Srinivasan, and Xudong Zhang. Battling the internet water army: Detection of hidden paid posters. In *International conference on Advances in Social Networks Analysis and Mining*, 2013.

[97] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *TKDD*, 29(3):626–688, 2015.

[98] Santiago Moiss Mola-Velasco. Wikipedia vandalism detection through machine learning: Feature review and new proposals - lab report for pan at clef 2010. In *CLEF*, 2010.

[99] B. Thomas Adler, Luca de Alfaro, and Ian Pye. Detecting wikipedia vandalism using wikitrust - lab report for PAN at CLEF 2010. In *CLEF*, 2010.

[100] Andrew G. West, Sampath Kannan, and Insup Lee. Detecting wikipedia vandalism via spatio-temporal analysis of revision metadata? In *EUROSEC*, pages 22–28, 2010.

[101] B. Thomas Adler, Luca de Alfaro, Santiago Moisés Mola-Velasco, Paolo Rosso, and Andrew G. West. Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. In *CICLing*, pages 277–288, 2011.

[102] Pascal Neis, Marcus Goetz, and Alexander Zipf. Towards automatic vandalism detection in openstreetmap. *ISPRS International Journal of Geo-Information*, 1(3):315–332, 2012.

[103] http://en.wikipedia.org/wiki/User:ClueBot_NG.

[104] http://en.wikipedia.org/wiki/Wikipedia:STiki.

[105] http://en.wikipedia.org/wiki/Wikipedia:Snuggle.

[106] Martin Potthast, Benno Stein, and Robert Gerling. Automatic vandalism detection in wikipedia. In *ECIR*, pages 663–668. 2008.

[107] https://www.mediawiki.org/wiki/API.

[108] http://beta.degreesofwikipedia.com/.

[109] Aaron Halfaker. http://datahub.io/dataset/english-wikipedia-reverts.

[110] Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. He says, she says: Conflict and coordination in wikipedia. In *SIGCHI*, pages 453–462, 2007.

[111] http://armstrong.cis.upenn.edu/stiki_api.php?

[112] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[113] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[114] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[115] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *SciPy*, June 2010.

[116] Don Fallis. A functional analysis of disinformation. *iConference*, 2014.

[117] Peter Hernon. Disinformation and misinformation through the Internet: Findings of an exploratory study. *Government Information Quarterly*, 12(2):133–139, 1995.

[118] Harry Frankfurt. On bullshit. *Raritan Quarterly Review*, 6(2):81–100, 1986.

[119] Nadia Khomami. Woman dies after taking 'diet pills' bought over internet. Website, 2015. http://www.theguardian.com/society/2015/apr/21/woman-dies-after-taking-diet-pills-bought-over-internet (accessed Oct. 16, 2015).

[120] Thomas Lavergne, Tanguy Urvoy, and François Yvon. Detecting fake content with relative entropy scoring. In *PAN*, 2008.

[121] Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

[122] Wikipedia. Balboa Creole French. Website, 2015. https://en.wikipedia.org/w/index.php?title=Wikipedia_talk:List_of_hoaxes_on_Wikipedia/Balboa_Creole_French&oldid=570091609 (accessed Oct. 16, 2015).

[123] R.J. DeNardo. *The Captain's Propensity: The Andromeda Incident II*. Strategic Book Publishing, 2013.

[124] M.S. McCormick. *Atheism and the Case Against Christ*. Prometheus Books, 2012.

[125] Kevin Morris. After a half-decade, massive Wikipedia hoax finally exposed. Website, 2013. http://www.dailydot.com/news/wikipedia-bicholim-conflict-hoax-deleted (accessed Oct. 16, 2015).

[126] Jimmy Wales. How frequent are Wikipedia hoaxes like the "Bicholim Conflict"? Website, 2013. https://www.quora.com/How-frequent-are-Wikipedia-hoaxes-like-the-Bicholim-Conflict (accessed Oct. 16, 2015).

[127] Wikipedia. Alan Mcilwraith. Website, 2015. https://en.wikipedia.org/w/index.php?title=Alan_Mcilwraith&oldid=682760877 (accessed Oct. 16, 2015).

[128] Wikipedia. Wikipedia Seigenthaler biography incident. Website, 2015. https://en.wikipedia.org/w/index.php?title=Wikipedia_Seigenthaler_biography_incident&oldid=677556119 (accessed Oct. 16, 2015).

[129] Wikimedia Foundation. Page view statistics for Wikimedia projects. Website, 2015. https://dumps.wikimedia.org/other/pagecounts-raw (accessed Oct. 16, 2015).

[130] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[131] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[132] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, pages 41–55, 1983.

[133] Jeffrey T Hancock. Digital deception. *Oxford handbook of internet psychology*, pages 289–301, 2007.

[134] Zhan Bu, Zhengyou Xia, and Jiandong Wang. A sock puppet detection algorithm on virtual spaces. *Knowledge-Based Systems*, 37:366–377, 2013.

[135] Dong Liu, Quanyuan Wu, Weihong Han, and Bin Zhou. Sockpuppet gang detection on social media sites. *Frontiers of Computer Science*, 10(1):124–135, 2016.

[136] Xueling Zheng, Yiu Ming Lai, Kam-Pui Chow, Lucas CK Hui, and Siu-Ming Yiu. Sockpuppet detection in online discussion forums. In *Proceedings of the Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2011.

[137] Wikipedia sockpuppet investigation policy. https://goo.gl/89ieoY. Accessed: 2016-10-24.

[138] Fredrik Johansson, Lisa Kaati, and Amendra Shrestha. Detecting multiple aliases in social media. In *International conference on Advances in Social Networks Analysis and Mining*, 2013.

[139] Tieyun Qian and Bing Liu. Identifying multiple userids of the same author. In *EMNLP*, 2013.

[140] James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

[141] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[142] Martin Potthast, Benno Stein, and Robert Gerling. Automatic vandalism detection in wikipedia. In *European Conference on Information Retrieval*, pages 663–668, 2008.

[143] HaiHai Yuan, Donald O Clifton, Phil Stone, and Herb H Blumberg. Positive and negative words: Their association with leadership talent and effectiveness. *The Psychologist-Manager Journal*, 4(2):199, 2000.

[144] Kuan-Ta Chen and Li-Wen Hong. User identification based on game-play activity patterns. In *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, 2007.

[145] John P Dickerson, Vadim Kagan, and VS Subrahmanian. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, 2014.

[146] Jeremy Blackburn, Ramanuja Simha, Nicolas Kourtellis, Xiang Zuo, Matei Ripeanu, John Skvoretz, and Adriana Iamnitchi. Branded with a scarlet c: cheaters in a gaming social network. In *Proceedings of the 21st International Conference on World Wide Web*, 2012.

[147] Shlomo Argamon and Shlomo Levitan. Measuring the usefulness of function words for authorship attribution. In *Proceedings of the 2005 ACH/ALLC Conference*, 2005.

[148] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. The slashdot zoo: mining a social network with negative edges. In *WWW*, pages 741–750, 2009.

[149] Silviu Maniu, Bogdan Cautis, and Talel Abdessalem. Building a signed network from interactions in wikipedia. In *DBSocial*, pages 19–24, 2011.

[150] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! In *ICWSM*, 2011.

[151] Ronen Feldman. Techniques and applications for sentiment analysis. *Commun. ACM*, 56(4):82–89, 2013.

[152] Phillip Bonacich and Paulette Lloyd. Calculating status with negative relations. *Social Networks*, 26(4):331 – 338, 2004.

[153] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, pages 641–650, 2010.

[154] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. Signed networks in social media. In *CHI*, pages 1361–1370, 2010.

[155] Aghaie A. Zolfaghar K. Mining trust and distrust relationships in social web applications. In *IEEE ICCP*, page 7380, 2010.

[156] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[157] Moshen Shahriari and Mahdi Jalili. Ranking nodes in signed social networks. *Social Netw. Analys. Mining*, 4(1), 2014.

[158] Philip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[159] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[160] Marc Najork, Hugo Zaragoza, and Michael J. Taylor. Hits on the web: how does it compare? In *SIGIR*, pages 471–478, 2007.

[161] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *ECML-PKDD*, 2015.

[162] Huan Sun, Alex Morales, and Xifeng Yan. Synthetic review spamming and defense. In *KDD*, 2013.

[163] Amir Fayazi, Kyumin Lee, James Caverlee, and Anna Squicciarini. Uncovering crowdsourced manipulation of online reviews. In *SIGIR*, 2015.

[164] Vlad Sandulescu and Martin Ester. Detecting singleton review spammers using semantic similarity. In *WWW*, 2015.

[165] Meng Jiang, Peng Cui, and Christos Faloutsos. Suspicious behavior detection: current trends and future directions. *IEEE Intelligent Systems*, 2016.

[166] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *RecSys*, 2007.

[167] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*, 2013.

[168] Paridhi Jain, Ponnurangam Kumaraguru, and Anupam Joshi. @ i seek'fb. me': Identifying users across multiple online social networks. In *Proceedings of the 22nd International Conference on World Wide Web*, 2013.

[169] Giuseppe Silvestri, Jie Yang, Alessandro Bozzon, and Andrea Tagarelli. Linking accounts across social networks: the case of stackoverflow, github and twitter. In *International Workshop on Knowledge Discovery on the WEB*, 2015.

[170] Reza Zafarani and Huan Liu. Connecting users across social media sites: a behavioral-modeling approach. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.