HAL open science

Anti-Abuse Protection of Online Social Networks using Machine Learning

Nour El-Mawass

▶ To cite this version:

Nour El-Mawass. Anti-Abuse Protection of Online Social Networks using Machine Learning. Artificial Intelligence [cs.AI]. Normandie Université, 2020. English. NNT : 2020NORMR094 . tel-03188653

HAL Id: tel-03188653

https://theses.hal.science/tel-03188653v1

Submitted on 2 Apr 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de « l'Université de Rouen Normandie »

Protection Anti-Abus de Réseaux Sociaux Numériques par Apprentissage Statistique

Présentée et soutenue par
Nour EL MAWASS

Thèse soutenue publiquement le 14 octobre 2020 devant le jury composé de		
Mme Armelle BRUN	Maître de conférences, HDR, LORIA, Université de Lorraine	Rapportrice
M. Osmar ZAIANE	Professeur, University of Alberta, Canada	Rapporteur
Mme Christine LARGERON	Professeur, LHC, Université de St-Etienne	Examinatrice
M. Babiga BIRREGAH	MCF, Université Technologique de Troyes, Institut Charles Delaunay	Examinateur
M. Alain RAKOTOMAMONJY	Professeur, LITIS, Université de Rouen	Examinateur
M. Paul HONEINE	Professeur, LITIS, Université de Rouen Normandie	Directeur de thèse
M. Laurent VERCOUTER	Professeur, LITIS, INSA Rouen Normandie	Codirecteur de thèse

Thèse dirigée par Paul HONEINE et Laurent VERCOUTER, laboratoire LITIS

UNIVERSITÉ DE ROUEN NORMANDIE

litis

Normandie Université

Thesis

Anti-Abuse Protection of Online Social Networks using Machine Learning

Nour EL-MAWASS

Directors:
Paul HONEINE
Laurent VERCOUTER

July 2020

Remerciements

Je remercie mes directeurs de thèse, Paul HONEINE et Laurent VERCOUTER, pour m'avoir accueilli au sein du LITIS et pour avoir facilité mon parcours par leur accompagnement, leur expertise et leurs conseils. Je leur suis énormément reconnaissante.

Je tiens à remercier également les membres du LITIS, et plus spécialement ceux de l'équipe MIND et DocApp, pour les moments partagés et les nombreuses discussions. Je remercie aussi Fabienne et Brigitte, pour leur soutien logistique et leur présence dès mon premier jour à l'université.

Mme Armelle BRUN et M. Osmar ZAIANE m'ont fait l'honneur d'être rapporteurs de ma thèse. Je les remercie pour le temps et l'attention qu'ils ont accordés à la lecture de ce manuscrit et les commentaires qu'ils ont rédigés. Je remercie aussi les examinateurs Mme Christine LARGERON, M. Babiga BIRREGAH et M. Alain RAKOTOMAMONJY pour avoir accepté de participer au Jury.

Enfin je ne peux manquer de mentionner ma famille, mes parents, mon mari et ma fille. Un simple merci n'est pas suffisant pour vous exprimer mon affection et ma gratitude.

Résumé

Au cours de la dernière décennie, la popularité incomparable des réseaux sociaux numériques s'est traduite par l'omniprésence des spammeurs sur ces plateformes. Cette présence a commencé par se manifester sous la forme de messages de publicité et d'arnaques traditionnels simples à identifier. Pourtant, elle s'est métamorphosée durant les dernières années, et couvre dorénavant de larges tentatives de manipulation qui sont assez importantes et beaucoup plus préoccupantes. Cet abus ciblé et largement automatisé des réseaux sociaux numériques réduit la crédibilité et l'utilité des informations diffusées sur ces plateformes.

Le problème de détection du spam social a été traditionnellement modélisé comme un problème de classification supervisée où l'objectif est de classer les comptes sociaux individuellement. Ce choix est problématique pour deux raisons. Tout d'abord, la nature dynamique du spam social rend les performances des systèmes supervisés difficiles à maintenir. En outre, la modélisation basée sur les caractéristiques (features) des comptes sociaux individuels ne prend pas en compte le contexte collusoire dans lequel les attaques sur les réseaux sociaux sont menées.

Pour maximiser leur efficacité et la visibilité de leur contenu, les spammeurs actent d'une manière qu'on peut décrire comme "synchronisée". Ainsi, même lorsque les spammeurs changent de caractéristiques, ils continuent à agir de manière collusoire, créant des liens entre les comptes complices. Ceci constitue un signal non supervisé qui est relativement facile à maintenir et difficile à contourner. Il est donc avantageux de trouver une mesure de similarité adaptée qui soit capable de capturer ce comportement collusoire.

Dans ce travail, nous proposons d'exprimer le problème de détection de spam social en termes probabilistes en utilisant le cadre des modèles graphiques non dirigés. Au lieu du paradigme de détection individuelle qui est couramment utilisé dans la littérature, nous cherchons à modéliser la tâche de classification comme une tâche d'inférence sur la probabilité jointe d'un graphe de variables. Dans ce contexte, les comptes sont représentés comme des variables aléatoires et la dépendance entre ces variables est représentée par un graphe. Cette expression probabiliste permet de modéliser l'incertitude inhérente aux systèmes de classification. Le graphe permet aussi d'exploiter la dépendance qui découle de la similitude induite par le comportement collusoire des spammeurs.

Nous proposons deux modèles graphiques: le Champs Aléatoire de Markov où l'inférence est effectuée par l'algorithme de Propagation des Convictions à Boucle, et le Champs Aléatoire Conditionnel, où on choisit d'utiliser l'algorithme du Tree

Reweighted Message Passing pour l'inférence et une fonction de perte qui minimise le risque empirique. Les deux modèles, évalués sur Twitter, montrent une augmentation des performances de classification par rapport aux classifieurs supervisés de la littérature. Le Champ Aléatoire Conditionnel offre de meilleures performances de classification par rapport au Champs Aléatoire de Markov. Il est aussi plus robuste face aux changements dans la distribution des caractéristiques des spammeurs.

Abstract

Over the last decade, the growing popularity of Online Social Networks has attracted a pervasive presence of social spammers. While this presence has started with spam advertising and common scams, the recent years have seen this escalate to the far more concerning mass manipulation attempts. This targeted and largely automated abuse of social platforms is risking the credibility and usefulness of the information disseminated on these platforms.

The social spam detection problem has been traditionally modeled as a supervised problem where the goal is to classify individual social accounts. This common choice is problematic for two reasons. First, the dynamic and adversarial nature of social spam makes the performance achieved by features-based supervised systems hard to maintain. Second, features-based modeling of individual social accounts discards the collusive context in which social attacks are increasingly undertaken.

Acting synchronously allows spammers to gain greater exposure and efficiently disseminate their content. Thus, even when spammers change their characteristics, they continue to act collusively, inevitably creating links between collusive spamming accounts. This constitutes an unsupervised signal that is relatively easy to maintain and hard to evade. It is therefore beneficial to find a suitable similarity measure that captures this collusive behavior.

Accordingly, we propose in this work to cast the social spam detection problem in probabilistic terms using the undirected graphical models framework. Instead of the individual detection paradigm that is commonly used in the literature, we aim to model the classification task as one of joint inference. In this context, accounts are represented as random variables and the dependency between these variables is encoded in a graphical structure. This probabilistic setting allows to model the uncertainty that is inherent to classification systems while simultaneously leveraging the dependency that flows from the similarity induced by the spammers collusive behavior.

We propose two graphical models: the Markov Random Field with inference performed via Loopy Belief Propagation, and the Conditional Random Field with a setting that is more adapted to the classification problem, namely by adopting the Tree Reweighted message passing algorithm for inference and a loss that minimizes the empirical risk. Both models, evaluated on Twitter, demonstrate an increase in classification performance compared to state-of-the-art supervised classifiers. Compared to the Markov Random Field, the proposed Conditional Random Field framework offers a better classification performance and a higher robustness to changes in spammers input distribution.

Contents

Remerciements **1**

Résumé **3**

Abstract **5**

Contents **6**

List of Figures **10**

List of Tables **12**

1 Introduction **13**
1.1 Research Context and Motivation 13
1.2 Research Questions 15
1.3 Research Objectives and Contributions 15
1.4 Thesis Outline 16

2 A Taxonomy and Overview of Social Spam **19**
2.1 Introduction 19
2.2 A Taxonomy of Social Spam and Spammers 20
2.2.1 General Notions of Online Social Networks 21
2.2.1.1 Users Content 21
2.2.1.2 Social Ties between Users 21
2.2.1.3 Interaction between Users 22
2.2.1.4 Special Operators 22
2.2.2 Social Spam Revenue Models 23
2.2.3 Spammers and Sybils on OSNs 23
2.2.3.1 Account's Origin 23
2.2.3.2 Account's Automation 24
2.2.4 A Taxonomy of Abusive Behavior on OSNs 24
2.2.4.1 Spam-as-a-service on OSNs 25
2.2.4.2 Abusive Content on OSNs 26
2.2.4.3 Spam Dissemination Channels 27
2.3 Data Collection and Labeling on Twitter 28

2.3.1 Data Collection Methods on Twitter 29
2.3.1.1 Twitter's Developers APIs 29
2.3.1.2 Accounts Sampling Approaches 30
2.3.2 Data Labeling of Twitter's Spam Datasets 31
2.3.2.1 Manual Labeling 31
2.3.2.2 Heuristics for Acquiring Malicious Labels 32
2.3.2.3 Heuristics for Acquiring Legitimate Labels 34
2.4 Detecting and controlling Spam on OSNs 35
2.4.1 Supervised Detection of Spam on OSNs 36
2.4.2 Graph-based Detection of Social Spam 36
2.4.3 Unsupervised Detection of Social Spam 38
2.4.4 Belief Propagation for Online Abuse Detection 38

3 Supervised Detection of Spammers on Twitter 43
3.1 Introduction 43
3.2 Modeling Accounts as Numerical Features on Twitter 45
3.2.1 General Profile Attributes 46
3.2.2 Social Network Features 46
3.2.3 Automation-related Features 48
3.2.4 Behavioral Features 48
3.2.5 Content-related Features 49
3.3 Empirical Evaluation of Twitter's Suspension Mechanisms 49
3.3.1 Compared Datasets 50
3.3.1.1 Checking for users suspension 51
3.3.1.2 Accounts labeling 51
3.3.2 Discussion 52
3.3.3 Conclusion 54
3.4 Detecting Spammers on Twitter: A Case Study 54
3.4.1 Collection and Labeling of a Ground-truth Spam Dataset . . . 55
3.4.2 Evaluation of State-of-the-art Detection Features 58
3.4.2.1 Statistical Account Features 58
3.4.2.2 Choosing Machine Learning Classification Models . 61
3.4.2.3 Evaluation on the Ground-truth Dataset 62
3.5 Conclusion 64

4 Social Spammers Detection using MRF 69
4.1 Introduction 69
4.2 Probabilistic Modeling of Spam Classification 71
4.2.1 An Informal Introduction to Probabilistic Graphical Models . 71
4.2.2 Formal Definition of the Spam Classification Task 73
4.3 Proposed MRF Solution 73
4.3.1 General Solution Design 74
4.3.2 Markov Random Field for Social Spammers Detection 75
4.3.2.1 MRF Potentials 75
4.3.2.2 Computing Marginal Probabilities by Loopy Belief Propagation 77

4.4 Constructing a Users-similarity Graph 78
4.4.1 Representing Similarity on OSNs 79
4.4.1.1 Similarity on Social Graphs 79
4.4.1.2 Similarity on Interaction Graphs 79
4.4.2 Proposed Content-based Similarity 80
4.4.2.1 Bipartite Users-Messages Graph 81
4.4.2.2 Users Similarity Graph 81
4.4.2.3 Twitter-Specific Considerations 82
4.4.2.4 Implementation 84
4.4.2.5 Computational Complexity of Graph Construction 86
4.5 Experimental Evaluation and Discussion 88
4.5.1 Experimental Setting 89
4.5.2 MRF Classification Results 90
4.5.3 Discussion and Generalization Insights 93
4.5.3.1 Modularity vs. Homophily 93
4.5.3.2 Effect of baseline recall and precision 94
4.5.3.3 Role of the edge potentials matrix 94
4.6 Conclusion 94

5 Spammers Detection with CRF 97
5.1 Introduction 97
5.1.1 CRF vs. MRF 98
5.1.2 Learning UGM parameters 99
5.2 A CRF framework for Social Spammers Detection 100
5.2.1 Problem Formulation 101
5.2.2 Proposed solution 101
5.2.2.1 Node Potentials 101
5.2.2.2 Edge Potentials 102
5.3 Inference and learning in Conditional Random Fields 103
5.3.1 Definition of Undirected Graphical Models 103
5.3.2 Parameters Learning in Undirected Graphical Models 105
5.3.3 Loss functions 105
5.3.3.1 MLE and the Log-Likeliohood Loss 105
5.3.3.2 ERM and the Clique Loss 106
5.3.4 Inference 107
5.3.4.1 Exact Variational Principle 107
5.3.4.2 Loopy Belief Propagation 107
5.3.4.3 Bethe free energy minimization 108
5.3.4.4 Tree-Reweighted Message Passing 108
5.4 Experimental Evaluation on Twitter 109
5.4.1 Graphical Models Implementation 109
5.4.2 Results and Discussion 109
5.4.2.1 Efficiency 111
5.4.2.2 Objective Loss 111
5.4.2.3 CRF parameters 111
5.5 Conclusion 112

6 Conclusion **113**

A Features extracted from Twitter accounts **117**

Bibliography **123**

List of Figures

2.1 A taxonomy of social spam on OSNs. 39
2.2 The authorization page of a free followers boosting application on Twitter showing the permissions the application gets when granted access to the account. 40
2.3 An example of trend-hijacking spam on Twitter. 40
2.4 An example of direct messages spam on Twitter. 41
2.5 A taxonomy of social spam datasets on Twitter. 41

3.1 A simplified partial overview of the supervised classification pipeline as applied to spam accounts detection on OSNs. 45
3.2 Different types of features extracted from a social account. 46
3.3 A screenshot of a compromised verified account posting a tweet containing a phishing link. 57
3.4 The CDF plots of the top 9 relevant features in the ground-truth dataset as selected by the mutual information method. 61
3.5 Distribution plots of features showcasing a dual spammer behavior. . . . 61
3.6 Distribution of probabilities computed by multiple classification models on the accounts represented by our set of features. 66
3.7 Distribution of probabilities computed by multiple classification models on the accounts represented by Benevenuto's set of features. 66
3.8 Distribution of probabilities computed by multiple classification models on the accounts represented by Stringhini's set of features. 67

4.1 A toy example showing how having a connection between two accounts alters beliefs about their class probabilities. 72
4.2 General architecture of the proposed system. 75
4.3 A graph of users illustrating node potentials defined over users and edge potentials defined over edges. 76
4.4 The edge potential matrix in (a) a symmetric special case, (b) an asymmetric case where inter-spammers and inter-legitimate connections are assigned different strengths. 77
4.5 Works based on the strong trust assumption, model the Sybils detection problem as two dense clusters of legitimate and Sybil accounts and assume that edges between these two clusters are attack edges. 80

4.6 Bipartite interaction graphs on YouTube (accounts commenting on videos) and Facebook (accounts liking pages). Dense bipartite clusters may indicate malicious activity. 80
4.7 Construction of the bipartite users-messages graph on a toy example. . . 81
4.8 A toy example showing the computation of similarity based on applications profile. 84
4.9 The similarity graph of connected users in the ground-truth dataset. Legitimate users are represented in green while spammers are shown in red. 85
4.10 Time required to expand a list of users into edges. 87
4.11 Time required to generate users edges. 87
4.12 Histogram of the distribution of identical texts in a trending topic. 88
4.13 Prior beliefs computed by the supervised classifiers over a cluster of spammers. Nodes predicted as legitimate users are represented in green, while red represents nodes predicted as spammers. 90
4.14 Loopy Belief Propagation illustrated on a cluster of 3 spammers. Input priors and potentials are shown on the left. The central frame shows the first and last iterations of belief propagation. The output frame shows posterior probabilities computed from the final values of messages. The algorithm in this instance converges after 4 iterations. 91
4.15 Performance gain of the MRF classification as a function of edge potentials for α ranging from 2 to 3.5. 92
4.16 Average absolute gain in performance as a function of edge potentials for $\alpha = 3.5$. Upper and lower limits correspond to the maximum and minimum gain at each value of w. 92

5.1 An example of two graphs having the same labels and different observations. 99
5.2 General architecture of the proposed system. 102
5.3 The evolution of loss as a function of the iterations of the optimization algorithm (average over 100 runs). 110
5.4 The F1-measure as a function of loss for the 100 runs of the learning algorithm. 111
5.5 The ratio $\phi_u(x = 1)/\phi_u(x = 0)$ of the node potential for the two possible states as a function of the node belief $y_1 = p(x = 1)$ for beliefs computed using the different possible configurations of supervised classification models. 112

List of Tables

2.1 Supervised spam detection approaches on Twitter: a classification by detection objective and detection granularity 37

3.1 Advanced features computed over an account's ego network. 47
3.2 Features capturing the social and content-posting behavior of a social account. 48
3.3 Features computed on an account's content (collection of tweets). 50
3.4 Suspension and deletion rates of accounts in seven Twitter datasets. Spammers samples are shown in red, legitimate samples are shown in green, and random samples are shown in blue. 53
3.5 Evolution of suspension rates in the "Streaming API sample" dataset between July 2018 and April 2020. 54
3.6 A snippet from the *vendor-purchased-2019* dataset on the bot repository. This illustrates the format of publicly available Twitter datasets. 56
3.7 Characteristics of the ground-truth dataset 56
3.8 Description of features used in this work 59
3.9 The classification confusion matrix. 62
3.10 Average classification performance of supervised classifiers on the ground-truth dataset. 63

4.1 Notations used for bipartite and similarity graphs. 82
4.2 Most used applications in the groundtruth dataset (in terms of the number of unique messages). 85
4.3 Processed texts and applications of the top five tweets in the dataset (in terms of number of users sharing the tweet). 86
4.4 Classification performance, evaluated over the test dataset, of the baseline supervised classifiers, the symmetric MRF classifier ($e^{w_0} = e^{w_3} = 0.9$ and $e^{w_1} = e^{w_2} = 0.1$) and the asymmetric MRF classifier ($w = 0.6, \alpha = 2.5$). 91

5.1 The edge potential matrix $\phi_{(u,v)}(z_u, z_v)$. 102
5.2 Classification performance of local supervised classifiers, the symmetric and asymmetric MRF models and the CRF-based model. 110

Chapter 1

Introduction

Chapter Contents

1.1 Research Context and Motivation 13
1.2 Research Questions 15
1.3 Research Objectives and Contributions 15
1.4 Thesis Outline 16

1.1 Research Context and Motivation

Spam is the "de facto" companion of new web technologies. Its presence is noticeable on a plethora of web services including search engines, email providers and online reviews platforms. Among these technologies, Online Social Networks (OSNs) represent an especially attractive target for online spammers. These platforms, with their growing number of users (2.45 billion and 330 million monthly active users on Facebook [1] and Twitter [2] respectively[1]), offer a wide and accessible global market for spam. The clickthrough rate of spam URLs on OSNs is reported to be an order of magnitude larger [3] than that on spam emails [4], making the economic incentive on OSNs significantly higher than on traditional media. These platforms also allow for a rich social setting and representation that is lacking from other platforms (e.g. email).

OSNs have a dual identity. They are both social networks and news dissemination platforms. The first facet requires a strong notion of trust, while the second an assumption of credibility and trustworthiness. The presence of illicit spambots on social networks undermines both assumptions. Through unwanted advertisement, unsolicited communication, bulk content generation, artificial popularity inflation, and bot-induced opinion manipulation, spam accounts on social networks interfere with the intended experience of the social platform on many levels. And although

[1] The numbers of monthly active users are those reported in the third quarter of 2019 for Facebook and the first quarter of 2019 for Twitter.

spam is traditionally discussed as a security risk, a greater risk can be formulated in terms of credibility and usability.

On the most basic level, the existence of bots degrades the experience of the social network user, replacing a human-to-human communication with an experience where profit-oriented accounts pollute the online sphere with insignificant, contextless or harmful content. On a larger scale, the presence of massive manipulation has been noted in online discussions surrounding political (e.g. the American and French presidential elections [5, 6, 7] and the Syrian war [8]) and economical (e.g. stock market inflation [9]) topics. This severely undermines the credibility and trustworthiness of data issued from OSNs, and therefore decreases its usability for analytics and decision making. Bots massively manipulate OSNs' audience by masquerading as real users. Manipulation often takes the form of direct voicing of opinions. Since it can equally have the goal of obstructing access to genuine information on OSNs, bots are also known to flood discussions with irrelevant content, thus effectively blocking users from the content they are seeking.

Detecting spam and abuse on OSNs is therefore vital for the proper functioning of these platforms. Manual reporting and verification remain a core pillar of OSNs' defense mechanisms. Online platforms deploy armies of thousands of human annotators to track and control the spread of illicit content. These tools, however, do not scale. With hundreds of millions of messages generated daily on OSNs, automatic detection is the only viable way to detect and quarantine abusive accounts and content.

In the research literature, the social spam detection problem has been traditionally cast as a supervised classification problem. This approach relies on building statistical classifiers of social accounts (or messages) based on features extracted from their profile, content, behavior and social network. Many early studies have shown that supervised classifiers were indeed able to yield high classification performance [10, 11, 12]. Later works [13, 14] have shown, however, that the supervised learning paradigm falls short in keeping up with the complex and ever-changing social spam characteristics. The dynamic and adversarial nature of social spam renders these classification systems obsolete.

This phenomenon is coined as spam evolution in the literature [14]. This is the process through which spammers change their characteristics and behavior either in response to the deployed detection systems or to the changing nature of their environment. As part of the population drifts away from the known pattern of spammers, the recall of machine learning systems is usually asymmetrically impacted [15]. We use this asymmetrical deterioration in performance to motivate a change in the perception of supervised systems. Instead of *detection*, they can be seen as tools for *discovering* spammers in the wild.

Spammers modus operandi offers another exploitable loophole against evolution. For an effective content dissemination, spammers usually act collusively. A high collusion is generally an indicator of coordinated abusive behavior. This direction has led to the development of unsupervised detection systems [16, 17, 18] that bypass the need to construct and maintain ground-truth datasets. Simply put, the unsupervised approach is a graph-based approach that treats the problem as a search of dense sub-graphs [16] or as a clustering problem [17]. By modeling the problem as a com-

munity detection problem, it offers a severe contrast with the majority of supervised works that model detection as a classification of individual accounts.

But while unsupervised systems are adept at detecting large clusters of abusive users[2], it has been shown that manipulation on online social networks can be achieved through a limited number of coordinated accounts[3] [9, 19].

1.2 Research Questions

This work relies on a main central assumption: "While spam accounts may partially evade supervised detection systems, they must act collusively, thus maintaining a connection between them". Unlike statistical features, which are typically easier to evade, we conjecture that changing graph-related characteristics is more expensive in the sense that acting individually would defy the purpose of reaching a larger audience or inflating content popularity. The graph forms then the constant factor in an otherwise changing equation.

The question therefore boils down to the following:

- Finding a fitting similarity that would uncover connections between spam accounts. It should also minimize connections between those accounts and the legitimate population.
- Modeling the task of detection in probabilistic terms. The model should take into account that predictions of the supervised models are biased and that, when an account is a spammer, connected accounts are likely to be spammers too.

In this work, we propose to meet these two requirements by bringing together the notion of similarity and the notion of prior predictions under a probabilistic graphical model framework. In their simplest forms, Markov Random Fields (MRF) can often be discussed with a straightforward understanding of Loopy Belief Propagation. A more advanced formulation would require a more advanced discussion of learning, inference and adequate loss functions.

1.3 Research Objectives and Contributions

We organize hereafter the main research objectives of this work.

1. Evaluating and characterizing the effect of spam evolution on the classification performance of state-of-the-art supervised systems.
2. Proposing a definition of similarity that effectively captures spammers collusion and ensures a high degree of class homophily between users deemed similar.

[2]The average clusters detected by Facebook's SynchroTrap [17] contains 1730 users with the threshold being set at 200 users.

[3]An example of this is the botnet analyzed in [8]. The described botnet contained 130 Twitter social bots and was extensively used for political propaganda.

3. Proposing an undirected graphical models framework that exploits the proposed notions of belief and similarity.
4. Assessing the effectiveness of undirected graphical models in mitigating the effects of spam evolution on the performance of supervised classifiers.

The work in this manuscript is communicated in the following articles.

Articles in Peer Reviewed Journals

Nour El-Mawass, Paul Honeine, and Laurent Vercouter, "SimilCatch: Enhanced social spammers detection on Twitter using Markov Random Fields," *Information Processing & Management*, vol. 57, no. 6, 2020.

Nour El-Mawass, Paul Honeine, and Laurent Vercouter, "Characterizing and Detecting Social Spam and Abuse on Twitter: A Survey," *ACM Computing Surveys. In preparation* .

Nour El-Mawass, Paul Honeine, and Laurent Vercouter, "Conditional Random Field for Detecting Sybils on Online Social Networks," *Computers & Security*, Elsevier. *In preparation*

Articles in Peer Reviewed International Conferences

Nour El-Mawass, Paul Honeine, and Laurent Vercouter, "Supervised Classification of Social Spammers using a Similarity-based Markov Random Field Approach," in *Proceedings of the 5th Multidisciplinary International Social Networks Conference - MISNC'18*. ACM Press, 2018.

Articles in Peer Reviewed National Conferences

Nour El-Mawass, Paul Honeine, and Laurent Vercouter, "Champ Aléatoire de Markov pour la Détection Supervisée des Comptes Malicieux sur Twitter," in *20ème Conférence d'Apprentissage automatique (CAp)*, Rouen, France, Jun. 2018.

1.4 Thesis Outline

In this introductory chapter, we have discussed social spam and briefly reviewed its impact on OSNs usability and credibility. We also motivated automated detection of social spam, underlining that static detection systems are routinely becoming obsolete. We have introduced the notion of discovery of spammers via sub-optimal supervised classifiers. We conjectured that a strong similarity measure between accounts can offer a robust constant against the continuous change in spammers statistical features. We also alluded to the role that undirected graphical models can play in maintaining the robustness of detection systems against spam evasion. The remaining of this manuscript is organized in the following chapters:

Chapter 2 - A Taxonomy and Overview of Social Spam organizes the existing literature on social spam detection into a well-defined taxonomy classifying spam

accounts and content and presents the methods used in the literature to collect and label ground-truth datasets. It also overviews and compares the main methodologies of the social spam detection literature.

Chapter 3 - Supervised Detection of Social Spammers on Twitter presents an extensive overview of the features used in the literature to model social accounts for the classification task. It also illustrates the heterogeneous nature of ground-truth social spam datasets on Twitter by comparing the suspension rates of existing spam datasets. Features evaluation is then undertaken on a Twitter ground-truth dataset that we collect and annotate specifically for this task. We evaluate and compare several of the main state-of-the-art supervised classification models in the setup established by the features and the collected dataset. We use the results of the classification to discuss spam evolution and evasion and its asymmetric impact on the performance of supervised classifiers.

Chapter 4 - Enhanced Social Spammers Detection on Twitter using Markov Random Fields introduces an enriched formulation of the classification problem, its input and expected output. This formulation takes into account prior predictions of other state-of-the-art systems and exploits similarity between social accounts. We also propose a classification framework based on the Markov Random Fields (MRF) formulation and apply loopy belief propagation to infer posterior predictions on social accounts classes. The proposed model is evaluated on the previously introduced Twitter dataset. Results show a significant increase in recall compared to the baseline established by state-of-the-art supervised classifiers.

Chapter 5 - Conditional Modeling of Spammers Detection with Conditional Random Fields introduces the Conditional Random Fields (CRF) framework that replaces the MRF framework as a solution to the social spammers detection problem. We discuss learning and inference in undirected graphical models and propose to use an alternative inference algorithm and loss function to the ones used in Chapter 4. The chapter also presents and discusses the experimental results of applying the CRF model to the Twitter dataset.

Chapter 6 - Conclusion concludes the thesis, summarizing its main results and implications and discussing future perspectives.

Chapter 2

A Taxonomy and Overview of Social Spam

Chapter Contents

2.1 Introduction 19
2.2 A Taxonomy of Social Spam and Spammers 20
2.2.1 General Notions of Online Social Networks 21
2.2.2 Social Spam Revenue Models 23
2.2.3 Spammers and Sybils on OSNs 23
2.2.4 A Taxonomy of Abusive Behavior on OSNs 24
2.3 Data Collection and Labeling on Twitter 28
2.3.1 Data Collection Methods on Twitter 29
2.3.2 Data Labeling of Twitter's Spam Datasets 31
2.4 Detecting and controlling Spam on OSNs 35
2.4.1 Supervised Detection of Spam on OSNs 36
2.4.2 Graph-based Detection of Social Spam 36
2.4.3 Unsupervised Detection of Social Spam 38
2.4.4 Belief Propagation for Online Abuse Detection 38

2.1 Introduction

The last decade has seen the mass adoption of online social networks and, with it, the emergence of social spam. Social spam is a general term used to describe spam and abusive behavior on OSNs. It has the distinct nature of being ambiguous, where ambiguity covers two facets of social spam: definition and identification. The two concepts are related but not mutually interchangeable. *Definition* refers to the act of enumerating the characteristics that describe a spam account or an abusive behavior, while *identification* is closely tied to the process of building ground-truth datasets. Clearly defining the spam concept cannot guarantee an easy ground-truth collection.

Similarly, having access to a ground-truth set of accounts[1] does not imply that one has a clear definition of the concept defined in his ground-truth set.

Throughout this chapter, we will endeavor to summarize and classify the literature on social spam detection on OSNs. The above discussion on *definition* vs. *identification* will motivate the first two sections of this chapter, while the last section will focus on the problem of building and evaluating social spam detection systems with or without a ground-truth dataset.

Compared to other OSNs, Twitter has an accessible Application Programming Interface (API) and a relatively tolerant policy towards moderate data collection and analysis. This explains why the number of publications targeting spam on this particular platform is significantly higher than those focusing on other similar platforms (e.g. Facebook [20, 17, 16] and YouTube [21]). The discussion in this chapter is general to spam on online social networks since the taxonomy and techniques apply to various OSNs. The bulk of the discussed work understandably focuses on Twitter since it is both the preferred testbed for spam research and the OSN we have chosen to evaluate our systems in the next chapters.

Chapter Organization and Overview. This chapter reviews three main aspects of social spam research. We namely organize the existing literature into a well-defined taxonomy classifying spam accounts and content, and discuss methods used by spammers to camouflage themselves (Section 2.2). We then present the methods used in the literature to collect data from online social networks and to construct ground-truth datasets and discuss the advantages and limitations of each method (Section 2.3). We finally overview the main methodologies of the social spam detection literature (Section 2.4).

2.2 A Taxonomy of Social Spam and Spammers

The first work on social spammers is that of Yardi et al. [22]. This work, which analyzes spam accounts posting to a trending music topic, paints a picture of simplistic accounts that are easily identifiable by a handful of statistical patterns. The works [10, 11, 23] that immediately follow offer a more detailed view of these accounts. With the exception of the work of Benevenuto et al. [10], these works continue to assume that a limited set of statistical characteristics (in the order of 5 to 7 features) is sufficient to differentiate between spam accounts and legitimate accounts.

Contemporary spamming accounts are remarkably advanced and complex. This is largely explained by the platform's underground market adopting a spam-as-a-service economy. The specialized set of services and tools allows end users to focus on their abusive content rather than the spam production chain and message dissemination technicalities. A distinction between spam accounts, abusive behavior and spam content thus emerges.

In the following, we introduce general notions of online social networks (Section 2.2.1) and describe social spam revenue models (section 2.2.2). We then give a

[1] An example of this is having access to a set of accounts suspended by Twitter. The *identification* of the ground-truth data does not imply that the concept of a suspended account is neatly *defined*.

classification of spam accounts (section 2.2.3), and detail the manifestations of abusive behavior they exhibit (section 2.2.4). This includes discussing the spam production pipeline as well as abusive content posting and dissemination. The diagram in Figure 2.1 presents a general taxonomy of abusive accounts, behavior and content on OSNs.

2.2.1 General Notions of Online Social Networks

Online Social Networks such as Facebook and Twitter are online platforms that allow their users to create and share content and to form social ties with other users. Both Facebook (created in 2004) and Twitter (created in 2006) are massively popular, with the number of monthly active users of Twitter (resp. Facebook) equal to 330 million (resp. 2.45 billion). The number of pieces of content shared daily on these platforms is equal to 500 million on Twitter and 4.75 billion on Facebook [24, 25]. Twitter's 2019 third quarter revenue is 823.7 million USD [26], 85% of which is advertisement-generated [27] and the company is valued at 24 billion USD (as of December 2019 [28]). Today, Twitter's population growth, its revenue model, stock value and potential investments are all strongly threatened by abusive activities and platform mistreatment [29]. In 2017, Twitter has reported a year-over-year 10-fold increase [30] in daily action taken towards abusive accounts. Over the last two years, the platform's pursuit of malicious and abusive activities has been distinctly more consistent and aggressive[2]. The massive problem of spam and bots on the platform, however, remains to be solved [32].

2.2.1.1 Users Content

Content on OSNs is organized into messages (posts on Facebook and tweets on Twitter). These messages may contain text (including URL(s)) and media items (images or videos). Unlike Facebook's free form content, Twitter has a more restricted microblogging model that limits tweets to 140 characters (doubled to 280 in 2017). Another fundamental difference between the two platforms is that accounts on Twitter are public by default, making generated content accessible to all users and search engines. This is in severe contrast with Facebook's access levels that generally differ depending on the relationship between two accounts. The difference may stem from Twitter being viewed first as a blogging service with a free information flow and second as a social network [33], while Facebook is mainly a social network. Both characteristics, together with the API of Twitter, heavily bias the bulk of existing social spam contributions towards Twitter.

2.2.1.2 Social Ties between Users

Social connections between users can be either unidirectional or bidirectional (mutual) depending on the platform. Facebook has a bidirectional friendship relation-

[2] According to a 2018 Twitter's blog [31], the platform reported a year-over-year increase of 214% of accounts removed for violating spam policies. The number of spam reports also dropped from an average of $25,000$ per day in March, to $17,000$ per day in May. In Q1 2018, more than $142,000$ API applications (responsible for more than 130 million low-quality and spammy tweets) were suspended.

ship: if user u is a friend of user v, v is also a friend of u. Following on Twitter, however, is not necessarily bidirectional: A user u can follow user v without v following u back. In this scenario, u is a "follower" of v and v is a "friend" (or alternatively a "followee") of u.

2.2.1.3 Interaction between Users

Interaction between users can take one of these forms:

- A reply from one user to a post by another user.
- A private message seen only by the sender and receiver.
- A re-posting of a user's content by another user. This is known as a "share" on Facebook and a "Retweet" on Twitter.
- A direct post from one user to another. This is illustrated by wall posts on Facebook and direct mentions on Twitter.
- An endorsement from a user to another user content. This is done via likes on Facebook and Twitter (previously known as favoriting on Twitter).

A user mentions another user (not necessarily a friend), by including the mentioned user screen name preceded by the "@" symbol in the tweet (e.g. *@cristiano*). The same convention also appears in a reply tweet with the difference that a reply contains a link to the original tweet to which it is related.

2.2.1.4 Special Operators

In addition to the mention symbol cited above, both platforms use special operators to aggregate and tie related texts and to offer a unified and compact format of messages. These include:

- Hashtags: A hashtag is a special text entity preceded by the "#" symbol (e.g. *#michael_jackson*). Including a hashtag in a tweet means linking it to all the tweets that contain the same hashtag. This allows communities to grow around hashtags, and allows access via search for all tweets containing the relevant hashtag. When a given hashtag becomes popular in a certain region, it becomes a "trend", and appears on the Twitter main page of users in that region[3].
- URL shorteners: This practice is common to both platforms and allows a compact presentation of URLs. This is especially useful on Twitter where the tweet size is limited. It is important to note that spammers often rely on additional URL shorteners to obfuscate their final landing pages and several layers of URL shorteners are commonly used to that goal [34].

[3]The same applies for any sequence of words that happens to gain popularity, regardless of whether these words are preceded by the hashtag symbol (e.g. *Barack Obama*).

2.2.2 Social Spam Revenue Models

Twitter's underground economy has two basic types of revenue models: spamvertising and spam-as-a-service.

The basic spamvertising revenue model is based, like advertisement using email spam, on maximizing click-through rates and diverting social users attentions to advertised products or services.

The second revenue model is based on selling underground tools including both content dissemination tools (e.g. accounts – optionally with a harvested base of followers – and custom applications) and credibility boosting tools (fake followers and fake likes/retweets) to interested social users. Current evidence shows that popularity inflation techniques are impacting both genuine and Sybil users [35], although it remains unclear how much of the fake increase in popularity is explicitly solicited by celebrity accounts. Note that although credibilities of both the profile and its content can be increased by the credibility boosting schemes (i.e., followers and likes/retweets), the impact of more followers is an increase in an account's overall credibility, while more interaction with content via retweets and likes results in an increased popularity for the content in question.

2.2.3 Spammers and Sybils on OSNs

While many works profile abusive accounts under the generic "spammer" description, a careful review of the literature will show that this term does not hold a unique definition. A specific work's definition of an abusive account, depends on the data collection and labeling techniques used to construct the ground-truth dataset. These will be discussed in details in Section 2.3. It is therefore possible to classify accounts by the types of abusive behavior they engage in or the abusive content they produce. Since these will be discussed amply in the following sections, we propose here a classification that depends on only two criteria: the account's origin and its level of automation.

2.2.3.1 Account's Origin

The account origin refers to the original ownership of the account at the time it was created. We distinguish here mainly between compromised accounts and Sybil accounts. Compromised accounts [36, 37] are initially legitimate accounts that have been compromised by spammers either partially[4] or completely[5]. Compromise can be temporary or permanent [38, 39]. Sybil accounts on the other hand are generally defined as fake accounts created solely for the goal of abusing the platform. They

[4]Partial compromise of an account is achieved when the holder of the account allows malicious applications to gain permissions to act on his behalf (e.g. by posting tweets or following other users). The original account holder maintains access to the account and application permissions can be revoked at any time.

[5]Complete compromise of an account is done through credentials stealing. After obtaining an account credentials, it is possible to change these credentials, thus blocking the original account owner from accessing his account and allowing the stealing party to gain full ownership of the account.

are often created in mass and have varying customizing degrees. There is further a distinction between:

- market accounts, namely accounts that are available for trading and that can be rented simultaneously to different unrelated spam campaigns.
- custom accounts, which are uniquely used by the spam campaign that operates them.

2.2.3.2 Account's Automation

Given that spammers need to target a large audience, sharing spam manually is time and effort consuming and is therefore ill-adapted to the spammers' goal. There exists spam that is generated through human-operated accounts, but this is usually done individually and on a small scale. Similarly, a great proportion of Twitter's content (e.g. digests and news feeds) is in fact generated using automated means. Since the infrastructure and API used to achieve automation are provided by the platform itself, automation cannot be strictly considered as an abusive behavior. This being said, spam on Twitter heavily uses automation for content generation and dissemination, making the two concepts (i.e., spam and automation) strongly correlated.

There are three levels of automation [40, 41, 14]: fully automated accounts or bots, human-operated accounts, and hybrid accounts where content and actions are governed by a combination of manual and automated mechanisms. These hybrid accounts are known in the literature as "cyberbots". These different shades of automation are shared across both legitimate and abusive accounts. In fact, bots on Twitter are not necessarily spammers[6], nor spammers are necessarily bots. It is safe to conjecture that spammers are mostly composed of bots accounts but the variety of automated activity on Twitter prevents us from considering every bot to be an abusive account.

2.2.4 A Taxonomy of Abusive Behavior on OSNs

Abusive behavior on OSNs is any behavior that an account or a collection of accounts exhibits in violation of the OSN's rules and regulations. In Section 2.2.2, we have presented two revenue models for social spam: spamvertising and spam-as-a-service. The utility of spam, however, is not measured solely by the revenue it returns. While some spam messages, such as those advertising products or services or directing users to external URLs, have clear economic incentives and an evident revenue model similar to that of email spam, other types of spam, such as opinion manipulation spam, may seem to lack direct economic profit at the first sight. The real economic profit, however, is achieved by the infrastructure facilitating this type of spam. This infrastructure abstracts away the complexity of spam generation by making available the set of tools, applications and accounts necessary to abuse the OSN and manipulate its users. The utility of opinion manipulation, e.g. in a political

[6]News bots, such as those discussed in [42], or bots that act as continuously updating digests for special themes, are examples of benign bots.

context, cannot be directly measured in monetary terms. It follows that abusive behavior does not neatly map to the two previously discussed revenue models, but can be classified into two categories: engaging in the spam-as-a-service market, namely through artificial credibility inflation behavior, and posting abusive content.

2.2.4.1 Spam-as-a-service on OSNs

The spam-as-a-service economy relies on tools provided by the OSN underground market to abstract the spam generation chain and support end users with the products and services they need to establish credibility, and massively manage content, accounts and interactions. The main products of the underground are fake followers and fake interaction in the form of likes and retweets.

- **Followers selling:** This is one of the leading monetized activities of the spam underground. Followers selling is an established business with different traders and product options. Followers are sold in packages where a predefined price is given to every bundle of followers depending on the number of sold accounts, their overall activity (silent followers are less pricey than followers engaging in automated tweeting/retweeting activity), their language or nationality, or whether they are real (compromised accounts of actual users) or fake (Sybil accounts often created in mass and using automated means to generate content and activity).

- **Retweet/like selling:** Similar to followers selling, the *retweet/like selling* service is used to boost the perceived popularity and credibility of a content. Depending on the bought option, a tweet is simultaneously liked (or retweeted) by a set of market accounts.

These services are used both inside the underground to boost spammers credibility and inflate their popularity and by interested customers that otherwise have no relationship to or knowledge of the underground logistics. In-depth and longitudinal studies of the followers selling business are presented in [43, 44, 45].

Note that the service discussed so far is a "premium" (paid) version of the free service offered to other accounts. The free service consists in offering normal accounts a bundle of followers and retweets to their messages in exchange for the malicious application being granted the permission to use the account for following other users and retweeting their content (see an authorization page of such an app in Figure 2.2). By giving these apps the permission they require, unknowing users become compromised and become themselves "market accounts", i.e., part of the traded accounts in the underground economy.

Since procuring an audience for their messages is crucial to ensure an adequate outreach for their content, harvesting followers has been a persisting concern for spammers. In addition to disseminating their own spam content, accounts with high followers count can be used to publish paid messages or can be sold to interested buyers. Followers collection techniques can also be used to procure a "normal" appearance to Sybil underground accounts. Besides buying followers through followers-

selling services as discussed above, spammers can also exploit "follow churn", a well-known technique for collecting followers, which we introduce below.

- **Aggressive following/unfollowing (follow churn):** A technique based on the human tendency to reciprocate in social relations, aggressive following is practiced by spammers in hope of getting the followed accounts to follow them back. This following behavior is performed randomly, inorganically and in bulk. Accounts that do not follow spammers back are unfollowed (usually after a short grace period) and even reciprocating accounts can be later unfollowed to allow spammers to repeat their scheme on new accounts[7]. This following/unfollowing behavior is in direct violation of Twitter's rules[8] and may result in the suspension of the offender account[9].

  Some spammers advertise that they will "follow back" any account that follows them (either in the *screen name* or the *about me* sections). Although used mainly by spammers, this technique can also be used by aggressive promoters and normal users [46], as empirical research on spammers connections has also shown that a substantial fraction of spammers' followers is formed by "social butterflies" [47]. These are legitimate accounts that reciprocate the follow relationship with any account that follows them without checking its background. They are on average more popular and influential than normal Twitter accounts, and sometimes belong to celebrities and public figures [48].

2.2.4.2 Abusive Content on OSNs

In the following, we present a general classification of abusive content encountered on OSNs:

- **Spamvertising**: Spamvertising, loosely defined as the process of referring to spammers-related websites in emails, blogs and social media, is a general umbrella of many spam activities. The definition can be extended to cover tweets advertising products and services on social networks, and includes the following examples:
  - Tweets advertising followers selling and retweeting services.
  - Commercial products and services advertisement (e.g. perfumes, pharmaceutical products, properties, etc...).
  - Links to online news websites, forums and shops.

[7]Twitter limits the number of profiles an account can follow to 5000 users. This threshold can only be bypassed when a suitable (undisclosed) number of followers is gathered to balance the followers to followees ratio. To continue following accounts, spammers therefore need to "make space" to their new targets. See Twitter troubleshooting: why can't I follow more accounts? `https://support.twitter.com/articles/66885`

[8]Twitter terms of service explicitly bans "aggressive following (accounts who follow or unfollow Twitter accounts in a bulk, aggressive, or indiscriminate manner)". See Twitter's rules: `https://support.twitter.com/articles/18311`

[9]Following rules and best practices: `https://support.twitter.com/articles/68916`

  - Referrals, i.e., requests to follow someone's account (such as an aspiring singer) on other social media such as YouTube and Instagram.

- **Opinion manipulation:** The orchestrated use of accounts, tweets and retweets to manipulate the public perception of a political or social subject has been documented in many works [9, 7, 49]. This tool is particularly useful when a party wants to dominate the public opinion in situations of political unrest, and controversial social issues [50, 51]. This type of spam is exceptionally difficult to spot as the content is generally context-relevant. A directly related topic is "fake news" that gained a lot of attention in recent literature [52, 53, 54].

- **Phishing and malware:** Perhaps the most known type of email spam, phishing is equally existing on social media and has the ability to cause great security risks. A phishing message contains a malicious URL that redirects to a phishing website or a malware-downloading page. Phishing and malware messages are usually detected in the literature using URL blacklisting services (e.g. Google Safe Browsing[10].).

- **Scams:** Like phishing, scam on social media is a shout-back to popular advance-fee scams on email (e.g. the famous Nigerian prince scam). The successful execution of a social scam requires the establishment of a communication channel between the scammer and the victim. This can be done using one of the following options available either in the abusive tweet or the scammer account: 1) a link to the scammer's website, 2) a contact option: an email, phone number, bank account, or 3) direct contact via mentions or private messages.

- **Content pollution:** This includes generating quotes, proverbs and normal texts or randomly retweeting genuine content. It is usually a mean to generate legitimate content for underground accounts to obfuscate their real nature. Content pollution can be equally used for opinion manipulation. Instead of engaging in propaganda, accounts can sometimes submerge genuine discussion with irrelevant content. By preventing normal users from accessing relevant content in a trending topic, it acts in a similar way to a DoS attack[11].

2.2.4.3 Spam Dissemination Channels

Spam dissemination channels are platform's conventions through which users of the online social network can be exposed to spam. Each dissemination channel targets a different part of the users base and has a different reaching power. The main channels used to propagate abusive content can be summarized as follows:

- **Search spam:** Search spam targets users that are searching for common or popular terms. Spam tweets containing the query terms are shown in the

[10] Google Safe Browsing API: `https://safebrowsing.google.com/`

[11] In computer networks, a DoS (Denial of Service) attack is an attack where the perpetrator attempts to block users' access to a service or resource by making it unavailable.

search results. Spam tweets of this type often have a text or a URL that are unrelated to the search terms. A subtype of search spam is collective attention spam [55] (e.g. hijacking trending topics on Twitter, where spammers hijack trending topics by including the topic hashtag in their tweets, see an example in Figure 2.3).

- **Mention spam:** This includes spam messages that target a particular Twitter account (and eventually anyone visiting this account's personal page) by including the *screen name* of the account (preceded by the "@" symbol) in the tweet text (e.g. *@justinbieber*). This type of spam is particularly directed at popular Twitter accounts to profit from the high visibility ensured by these accounts' audience, but mention spam can also target normal or random users.

- **Direct Messages (DM) spam:** In direct messages spam, a spammer sends an unsolicited private message to the target accounts containing an undesired or potentially malicious content. Direct messages spam is hard to detect and measure by researchers with normal-level access, since it consists of private information shared only between the sender and the receiver, and that is only directly accessible to the OSN provider. Figure 2.4 shows an example of DM spam.

- **Followers-targeted spam:** This comprises spam messages that are published on the spammer's own page and can only be seen by his followers or page visitors. For this technique to be effective, the spammer in question should have harvested a large base of *real* followers using one of the followers acquiring methods described above.

2.3 Data Collection and Labeling on Twitter

Data collection consists in collecting messages and users profiles by connecting to the OSN's Application Programming Interface (API). Compared to other OSNs (e.g. Facebook), Twitter has the unique position of providing an accessible API. Twitter profiles are also public by default[12], meaning that their tweets can be seen by anyone on Twitter and consequently public profiles are also accessible to developers via the API. Sharing research datasets collected from Twitter has been severely regulated and restricted in the last few years [56], with direct effects on research reproducibility, but the ease of collection ensures that research based on Twitter's data continues to thrive.

An overview of the social spam data collection on Twitter is shown in the taxonomy in Figure 2.5. In the following, we will start by laying the foundations of data collection on Twitter. While the discussion is closely tied to the Twitter ecosystem, the general concepts are applicable to data collection on OSNs in general. We then discuss approaches used in the literature to label the collected dataset. Note that this distinction between collection and labeling is artificial. While these two steps are usually considered separate in the data mining pipeline, the constraints of the

[12] According to a comprehensive study [47], private profiles form 8% of the Twitter's population.

social spam domain often dictate that they merge into a single step. Thus, many of the labeling techniques discussed below can also be considered as data collection techniques.

2.3.1 Data Collection Methods on Twitter

The distinction between collection methods is fundamentally a distinction in accounts *sampling*. We first introduce technical aspects related to the Twitter API before discussing accounts sampling approaches.

2.3.1.1 Twitter's Developers APIs

Twitter provides an API that allows programmatic access to the data and automatic actions on behalf of the accounts. The communication is done via get and post requests and responses.

Format of the received response: The received information is a list of json-encoded objects with a rich information structure that includes not only the tweet or profile associated information available via the user interface, but also information that is only accessible by connecting to the API (e.g. users identifiers and until recently the device or application used to post a tweet). Both users and tweets have unique non-reusable and unchangeable numerical identifiers that are assigned to them upon creation. The identifiers are generated in an ascending order so that an identifier with a larger value indicates a more recently created account or tweet. In the following, we will describe the two main developers' APIs: the streaming API and the search API.

The streaming API: The streaming API[13] delivers two types of real-time data streams: the sampled stream and the filtered stream. The sampled stream, as its name indicates, delivers a real-time stream representing a 1% sample of Twitter's global stream. The filtered stream, on the other hand, delivers a sample of tweets (up to a maximum of 1% of the total stream) corresponding to a developer-defined query (a set of tracked keywords or followed users). The current free-access level to the streaming API replaces the more comprehensive access levels previously provided by Twitter (such as the elevated gardenhose access representing 10% of the total stream, and the complete firehose access representing 100% of the total stream). Enterprise-level access is nowadays managed by Gnip[14], Twitter's partner company that offers multiple options for a customized elevated access to Twitter's data.

The Search API: The search API[15] is the rate-limited[16] REST API allowing developers to customize their queries by "conducting singular searches and reading user

[13]Twitter's Streaming API: `https://dev.twitter.com/streaming/overview`.
[14]Gnip Twitter Access: `https://gnip.com/sources/twitter/`.
[15]Twitter's Rest API: `https://dev.twitter.com/rest/public`.
[16]Rate limiting on Twitter: `https://dev.twitter.com/rest/public/rate-limiting`.

profile information". It also allows applications with the right set of permissions to execute actions on behalf of the user, e.g. following other users and managing content (tweeting, deleting), as well as interacting with other users (sending direct messages, retweeting and liking other users' tweets). Depending on the type of the queried resource, the number of API calls allowed per a fifteen minutes window varies between 15 and 1500 queries[17]. Whitelisted access, now discontinued, consists in elevating the rate-limit of one or several whitelisted IP address to $20,000$ API calls per hour, therefore allowing the collection of datasets that are considerably larger and more comprehensive than datasets obtained through normal-level access.

2.3.1.2 Accounts Sampling Approaches

There exist different methods to collect data on OSNs. We will explain and compare each of these methods and discuss their advantages and limitations. The main data collection methods encountered in the literature are:

- Brute force accounts collection.
- Direct graph sampling.
- Random sampling of active users.
- Honeypot traps.

Brute force accounts collection: Brute force data collection requires a special elevated access to Twitter's infrastructure and dozens of whitelisted IP addresses working over a considerable collection period. The process described in the literature [47] consists of individually checking and collecting the account information of every numerical identifier starting from the lowest to the highest known identifier. Analysis and findings based on the dataset yielded by brute force access are of unmatched influence and significance, but are impossible to replicate as neither the collected dataset nor the collection method are available for researchers.

Direct graph sampling: This collection method starts from an initial seed of randomly chosen accounts and expands the list of collected accounts by following the social networks (followers or followees) of these users up to a given degree [22] often using a Breadth-first approach. The characteristics and representativeness of the collected nodes are discussed in [57].

Random sampling of active users: This is a fast and simple collection method that can quickly amass a large volume of users [58]. It consists in sampling accounts that are tweeting in Twitter's public timeline [23, 11] (now discontinued), in trending hashtags [22], or via the streaming API [59]. Although the goal is to randomly sample users, the method actually samples tweets posted during a given observation period, and later associates each tweet with a user ID. It follows that only users active during

[17] Limits of each query: `https://dev.twitter.com/rest/public/rate-limits`.

the observation period are sampled. The representativeness of the obtained datasets is also questionable since by sampling tweets (as opposed to accounts), the method oversamples frequent posters [59].

Honeypot traps: By creating honeypot accounts that mimic normal OSN users and integrate into the social network, researchers can trap spammers by luring them into following or interacting with these accounts. Since the honeypot method can be jointly considered as a collection and labeling method, we will discuss it in details in the next section.

2.3.2 Data Labeling of Twitter's Spam Datasets

Labeling refers to the act of assigning a class or label to a data instance in order to build a labeled dataset. In the current case, this means assigning either a spammer or a legitimate label to the labeled account. The labeled dataset is then used either to analyze spam on Twitter or to build and evaluate spam detection systems. Labeling is usually a human-intensive job that requires an expert human annotator and is therefore time-consuming and hard to scale. This explains why even when massive datasets are available, manually labeled datasets are modestly sized. It also explains why it is easier and more common to recur to heuristics that automatically label instances.

Ground-truth data used in social spam research is usually built via various labeling methods that are based on different empirical assumptions. This creates heterogeneously labeled datasets that are hard to compare against each other and are often limited to only a part of the studied problem. In the following, we will discuss the merits and deficiencies of each of these labeling approaches. We will namely discuss manual labeling as well as heuristics for acquiring legitimate and malicious labels. Note that heuristics-based labeling yields only one class of users, and thus needs to be complemented with labeled instances from the other class.

2.3.2.1 Manual Labeling

Unlike other labeling methods that use heuristics, curated blacklists, or internal signals from the Twitter's platform, manual labeling solely relies on the judgment of the human expert charged with annotating the collected data. Manual labeling is time and resource-consuming, and tends to produce datasets that are considerably smaller than datasets annotated through other labeling methods. And although the annotation process is completely controlled by the researchers undertaking the analysis, the resulting labels cannot be expected to form a universal ground-truth, since no spam definition is universal. Moreover, even when a definition is specifically created for the sake of the annotation process, it is common for human annotators to disagree on some of the labeled entities. Like other applications requiring human input, several works have framed the task of labeling accounts or messages on Twitter as a series of message labeling tasks [10, 60, 61, 62]. These tasks can be thus framed as "microtasks" and adapted for crowdworkers on crowdsourcing platforms such as Amazon Mechanical Turk.

2.3.2.2 Heuristics for Acquiring Malicious Labels

While heuristics discussed below offer a powerful way to scale the labeling process, they all have limitations. A particularly problematic aspect is the skewed representation of the spammers population as every heuristic focuses on one aspect of spam. Even more problematic is the assumption that a given heuristic is both sufficient and necessary for a tweet or an account to be a spam or a spammer. In other terms, everything not detected by the heuristic is considered legitimate. This assumption can potentially misguide any further analysis or results based on the obtained labels.

Honeypot traps: The honeypot approach consists of creating social honeypots, namely artificial accounts that often mimic the average user of the studied social network [11, 63, 64], and using them to attract and trap spammers. Compared to other collection methods, social honeypots have the advantage of continuously and autonomously collecting up-to-date evidence on content polluters from the social network [64]. Although elegant and simple, this collection/labeling method is problematic for the following reasons:

1. *Observation duration:* This approach requires a long duration (spanning months) of possibly inactive observation before a satisfactory users database is built.

2. *Sampling bias:* The resulting spammers dataset is biased toward spammers that actively follow other users. Randomly following other users in the hope of a follow back is only one of many techniques the spammers use to harvest followers.

3. *Need for human verification:* Since honeypot accounts appear legitimate, benign users may follow them for non-malicious reasons[18] such as the desire of being followed back, or a genuine desire to connect with the account [11]. The resulting dataset, therefore, cannot be used before being verified by a human to filter out legitimate accounts.

Labeling based on URLs blacklists: Approaches using URL blacklisting services to label spam on Twitter adopt the classic definition of spam defined as tweets containing blacklisted URLs [34, 67, 68, 69]. The safety of the URL is assessed through services such as Google Safe Browsing[19] and phishTank[20]. The tweeting account is considered a spammer if more than a threshold of its content (e.g. 10%) is flagged by these safe browsing services. This often offers a very skewed view of the spammers

[18]There are instances where benign users either follow [65] or interact with artificial accounts with genuine interest [66]. In the social experiment described in [66], authors set up a bot with no network of trust, no profile information, and no intention of reproducing a human-like behavior and let it interact with users. The experiment results show that, using simple automated techniques, the bot is able to become very relevant to the network, i.e., both top popular and influential.

[19]Google's Safe Browsing API: `https://developers.google.com/safe-browsing/?hl=en`.

[20]The PhishTank website: `https://www.phishtank.com/`.

scenery on Twitter. Even when only targeting spamvertising, this method yields a non-representative fraction of spamvertisers as communication can be established directly with the advertising account or via a phone number or email mentioned in the tweet text.

The main drawbacks of datasets collected and labeled using this definition are the following:

1. **A limited definition of spam:** As mentioned above, a blacklisted URL is not a necessary condition for spam. Tweets containing URLs form a limited fraction of spam tweets. Messages advertising followers selling services for example usually do not contain URLs. In addition, URLs contained in spamvertising tweets often redirect towards online stores (e.g. *amazon.com*) that are not flagged by blacklisting services, which means that these spam messages, although they contain URLs, are also not detected by this labeling approach.

2. **Blacklists update delays:** Given the delay between the time a URL is used on social media and the time it first appears on blacklists, it is possible that part of malicious tweets is not flagged by blacklists. To maximize the accuracy of the labeling method, blacklists should be checked repeatedly after a URL appears on social media before assigning a label.

Labeling based on suspended accounts: This approach defines spammers as accounts that are suspended by Twitter. In other terms, the definition is based on reverse engineering the platform's definition of malicious behavior after-the-fact. It has been used to study and uncover characteristics of the suspended population [70]. Although it provides a gold truth, it has several limitations, the least of them is the lack of a clear human-interpretable definition of what suspension means. More importantly, suspension on Twitter, particularly through massive purges, is a business act that involves weighting utility and minimizing false positives. This results in considerable delays and an ambiguous mapping between the suspended and spammer concepts:

- *Considerable delays:* While we do not have information on the internal mechanisms of Twitter's detection systems, suspension is only the tip of the iceberg. Twitter purges of spammers are infrequent and wide-spaced, as they impact the size of the population and have been known to significantly decrease the number of followers of celebrity profiles[21]. Consequently, unless suspension is done manually, a considerable delay is often noted between the time an account goes "rogue" and the time it gets suspended [70].

- *Not all malicious accounts are suspended:* Again due to business reasons, non-suspended accounts are not necessarily legitimate even if they have been active for a long time. The dynamics of suspension on Twitter can ultimately be

[21]The most recent large-scale bot purge on Twitter was conducted in July 2018 [71], and had a noticeable impact on the number of followers of many celebrities [72]. The purge lead to the removal of more than 9 million previously identified and locked accounts.

discussed in a game-theoretic context where suspension is a signaling act that can be exploited by spammers. Thus, it may be speculated that not suspending some accounts, particularly the less damaging ones, can be of greater utility in the larger spam evolution picture if it means less information leaked on the internal detection system[22].

- *Suspended accounts are not necessarily spammers:* Suspended accounts can be legitimate accounts suspended for non-spam related reasons. The distinction is not possible without manually reviewing the content and profiles of suspended accounts. Manual reviewing is needed because Twitter does not make public the reasons for suspension nor does it encourage or endorse the results of research based on suspended accounts [32].

To study the gap between malicious accounts and suspended accounts, we checked the current status of spamming accounts in three social spam datasets, namely, the honeypot dataset [63], the 1k-10k dataset [13] and the hashtag-hijackers dataset [73]. A considerable part of the malicious accounts present in these datasets still exists on the OSN. The details are discussed in Section 3.3.

Labeling based on reported accounts: Labeling based on reported accounts is similar to labeling based on suspended accounts, except that it uses identifiers of accounts that are reported by the Twitter community using tweets directed toward the official *@spam* account. Since some accounts can be maliciously or erroneously reported, the collected usernames should be manually verified before inclusion in the labeled dataset.

Dictionary-based labeling: Dictionary-based labeling often targets specialized spam (e.g. pharmaceutical spam), and may use spam dictionaries compiled from other domains (e.g. email) to detect and label spam content on OSN (usually through direct searching and matching of the dictionary keywords over the OSN messages). A recent example is the Pr0n dataset [74] that collects accounts marketing adult dating sites on Twitter. The collection algorithm starts from a seed of known spammers and crawls their ego networks looking for accounts displaying similar phrases and patterns.

2.3.2.3 Heuristics for Acquiring Legitimate Labels

Labeling based on reliable accounts: In the same spirit of labeling based on suspended and reported accounts, reliable accounts (e.g. accounts officially verified by Twitter, celebrities) can be used as examples of legitimate behavior on Twitter. The yielded dataset, however, cannot be considered as a representative sample of legitimate accounts, since the collected accounts have characteristics that are widely different from normal accounts.

[22] Alternatives to suspending accounts include *"reducing the visibility of suspicious accounts in Tweet and account metrics"* as reported by an 2018 Twitter's blog post [31].

Random sampling of accounts: Based on empirical studies and reports estimating the spam on OSN to form a minor percentage of the total content[23], some works conveniently propose to directly label randomly sampled accounts as legitimate accounts. This method is often used in conjunction with other collection and labeling methods to balance datasets that are mainly representative of spam. The assumption when using this labeling method is that the error induced by the labeling process is acceptable for the purposes of the undertaken analysis or application.

2.4 Detecting and controlling Spam on OSNs

The literature covered so far mainly tackles empirical analysis of social spam datasets. We next cover works geared towards solving the social spam problem. The vast majority of these works implicitly assume that the problem boils down to detection of abusive accounts and behavior. Another much less explored approach attempts to alleviate the problem by imposing design choices that ultimately decrease the perceived utility of abuse on the OSN platform (or alternatively increase the cost of engaging in collusive behavior). This approach thus adopts prevention rather than detection. The most notable example is *CollusionRank* [48], a trust-based PageRank-like social ranking system that assigns influence scores to users, penalizing those that link to spammers. The main aim of this system is to deter "social capitalists" (or influencers), from engaging in non-discriminant link farming (follow churn) that the study found to be an important catalyst for boosting the credibility of spam accounts on Twitter.

In the following, we focus on the spam detection approach. We propose to classify related work according to three criteria: detection objective, detection granularity and methodology.

Detection Objective. We indirectly referred to the detection objective in the social spam taxonomy we proposed in Section 2.2. We define the objective of the detection as the particular instance of the social platform the detection system wishes to identify or to label as abusive. There are three distinct instances found in the literature: the social account, the social post (e.g. a tweet), and the URL. The first instance refers to a single profile on the platform, while a post represents the atomic unit of content. A URL is also an instance of content, but unlike the first two, it is not an atomic component of the platform. It can be found in text-based components, such as posts and about-me sections.

Detection Granularity. For all of these distinct instances, the problem can equally be defined in terms of detection granularity. In this regard, detection can target individual instances or a community of instances. The latter can be referred to as collective detection. Let us take the example of accounts detection: in the individual detection case, the system assigns a label to each individual account. In collective detection, on the other hand, a unique label is assigned to each *community* of accounts. The mapping from accounts to communities is either predefined in in-

[23] Early estimates are around 4% [10] while more recent research places the numbers between 9% and 15% [75].

put (e.g. in supervised systems) or is part of the output of the detection system (e.g. in most of the unsupervised systems).

Detection Methodology. The remaining of this section will focus on discussing works grouped by methodology. By far the most used one is the supervised classification paradigm. Other discussed methodologies include graph-based detection, detection by loopy belief propagation and unsupervised classification.

2.4.1 Supervised Detection of Spam on OSNs

Supervised detection is understandably the dominant approach in the social spam detection literature. Based on the supervised classification paradigm, this approach models the objective of the detection system (e.g. accounts, messages or URLs) as a features vector and trains a supervised classification model on pairs of instances and labels. This definition is simple and computationally tractable. As features are generally defined per-instance, the computational cost of computing features is proportional to the number of classified instances.

Starting from 2010, many works focused on characterizing and identifying spammers using supervised learning tools and models [10, 11, 12, 63]. Table 2.1 provides an extensive list of the works published in the supervised literature. These works are classified according to the two main criteria we outlined in section 2.4, namely the detection objective and the detection granularity. The majority of the cited papers consider the spam detection problem to be an account classification problem, with a specific focus on individual detection of general spam accounts. Different formulations can be obtained by considering specific types of abusive accounts, e.g. fake followers [76, 77], or attempting to distinguish robotic from human accounts [41, 40, 62, 78, 79].

It is also possible to define the task as one of community classification (using community-based features) [80] or by performing classification on content. Classification on content can be done on the tweet level [81, 82, 83] or on the URL level [69, 84, 85, 34, 86].

Detecting URLs, as opposed to detecting accounts and messages, allows to tie the social spam detection problem to the existing search and URL spam problem by exploiting the features of the landing webpages of URLs found on the OSN.

The main contributions of these works can be formulated in terms of "features engineering". The proposed (and sometimes overlapping) sets of features are extracted from users' profiles, content, behavior and social network to characterize and identify spammers. We discuss these features in more details in Chapter 3.

2.4.2 Graph-based Detection of Social Spam

In contrast to the previously discussed approaches, which constitute the bulk of the community contributions, a more recent paradigm is centered around the graphical representation of the problem by exploiting a major loophole in the spam strategy. The guiding assumption of this paradigm is that, in order to be effective, malicious attacks need to be at least loosely coordinated or synchronized. This results in Sybil accounts being linked, either through the social graph structure or through some form

Table 2.1: Supervised spam detection approaches on Twitter: a classification by detection objective and detection granularity

Detection objective	**Detection granularity**	Works
Accounts	Individual	General [10, 11, 87, 23, 63, 64, 12, 13, 88, 20, 89, 90, 62, 91] Bots [40, 41, 62, 78, 79] Fake followers [76, 77]
	Collective	[80]
Messages	Individual	[10, 92, 55, 93, 81, 82, 83]
	Collective	[94, 86]
URLs	Individual	[34, 85, 86]
	Collective	[69, 84]
Co-detection (accounts and messages)	Collective	[95]

of similarity. This assumption is used to construct what can be called, depending on the application, a social [96, 97], interaction [18, 16] or similarity graph [17, 98].

Early works (e.g. SybilLimit [96], SybilInfer [99] and SybilGuard [97]) share the assumption that Sybil accounts form one community concentrated around a trusted node and leverage community detection algorithms, mainly through random walking to detect Sybil communities [100]. Intra-community links are considered strong-trust links, while inter-communities links are considered attack links. These works are unanimously evaluated on the who-connects-to-whom graph (who-follows-whom on Twitter), which is commonly referred to as the *social graph*. They are built with the assumption of strong-trust between connected nodes, which holds better on peer-to-peer networks than on OSNs [48, 101].

In more recent contributions, the social graph is replaced by an interaction graph (e.g. Facebook users liking pages in CopyCatch [16], and YouTube users commenting on videos in Leas [18]). When interaction is measured on many levels, the interaction graph can be transformed into a similarity graph (e.g. in Facebook's SynchroTrap [17]). Detection is executed either by means of graph clustering (or cutting) (e.g. SynchroTrap [17] and Leas [18]), or is modeled as a search for abnormally dense subgraphs (e.g. CopyCatch [16]). Note that both graph construction and graph search are computationally costly with the number of operations exponential in the number of users.

Some of the mentioned graph-based works are completely unsupervised [16, 17], while some, especially works based on graph cutting, assume the presence of at least one label to help associate the identified clusters (e.g. SybilInfer) [99]. There are also works that focus on localized clustering around known seeds of Sybil accounts to execute efficient detection (e.g. Leas [18]).

2.4.3 Unsupervised Detection of Social Spam

Most of the unsupervised detection approaches are graph-based approaches that either aim at clustering accounts or at identifying dense interaction subgraphs. The assumption is that spam accounts are connected in a way that is captured by the underlying social or interaction graph. These graphs form the basis of the approaches discussed above. There exists however, other unsupervised approaches that are based on individual modeling of accounts. These include Compa [37, 36], a system that identifies compromised accounts by the way of anomaly detection. Compa models each user as a statistical profile and flags anomalous messages as messages sent while the account is compromised. Another notable example is the work of Cresci et al. on DNA fingerprinting [102]. This approach assumes that collusive spammers have similar activity dynamics and proposes a DNA-like alphabet to model account's activity. Spammers are then detected by clustering similar DNA profiles.

2.4.4 Belief Propagation for Online Abuse Detection

Loopy Belief Propagation has been predominantly defined in the context of connected social accounts, where either labels are known for a part of the accounts or a belief can be defined for some or all of them. It has been mostly defined in association with a Markov Random Field although there exists exceptions where the algorithm is used without an undirected graphical model context (e.g. Polonium [103]).

This approach relies on propagating beliefs over a graph of accounts with the assumption of similarity between connected accounts. Early applications in the context of anomaly detection on online platforms include Netprobe [104] and FraudEagle [105, 106], which target fraudulent accounts on online markets (e.g. Amazon and eBay) and review fraud, respectively. A notable example in the domain of malware detection is Semantic Norton's Polonium system [103] which implements belief propagation over a large-scale bipartite graph of machines and files. Machines are assigned a proprietary reputation belief and the belief propagation helps identify malware files.

Despite some similarities with the problem of online social spam detection, the context and formulation of the aforementioned models are quite different, making it impossible to transfer them directly to the setting of social spam detection. Applications of Markov Random Field (MRF) to spam detection have followed the traditional model of earlier graph-based approaches discussed in Section 2.4.2 (e.g. SybilGuard [97] and SybilLimit [96]) by basing the users graph on the social structure of the network (the who-follows-whom graph). SybilBelief [107] is a system that propagates known labels of users over the social structure using an MRF model. The system is tested on synthetic and real-world social graphs including the social graph of Facebook. SybilFrame [108] is based on a similar idea but uses a probabilistic representation of users based on their perceived labels. The proposed system is evaluated on synthetic data and the social structure of Twitter. SybilBelief and SybilGuard are direct extensions of the established graph-based detection community, which traditionally bases detection on the social structure network with the assumption that links between users are based on a relationship of trust.

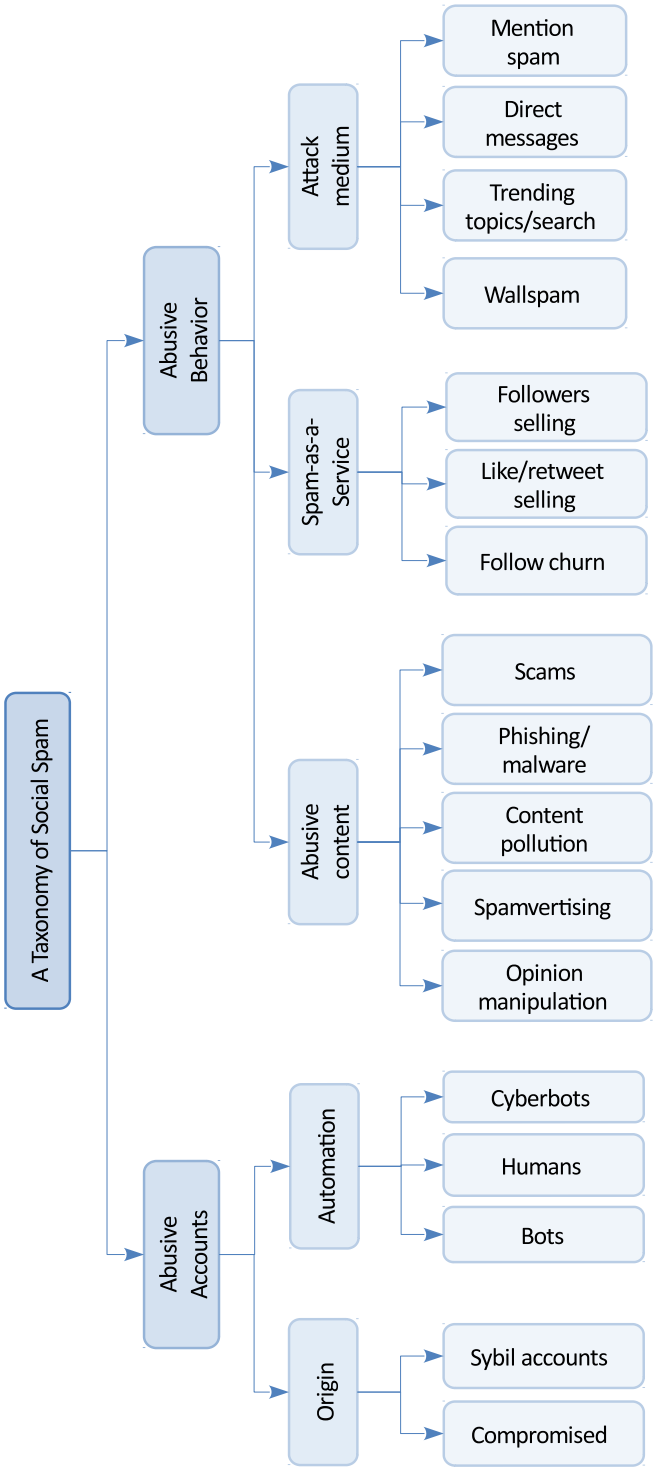

Figure 2.1: A taxonomy of social spam on OSNs.

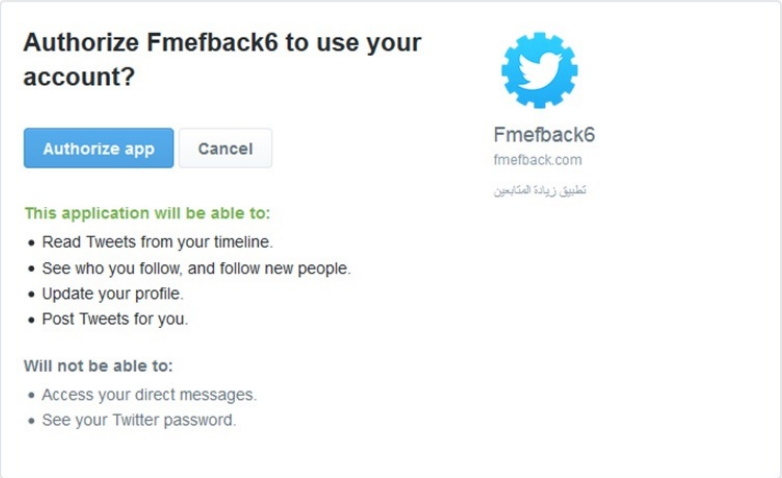

Figure 2.2: The authorization page of a free followers boosting application on Twitter showing the permissions the application gets when granted access to the account.

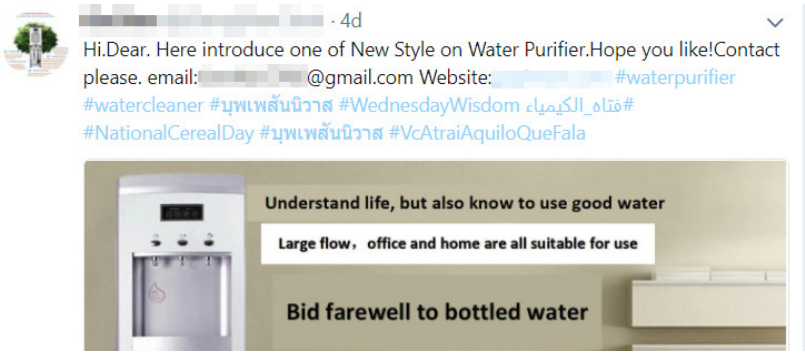

Figure 2.3: An example of trend-hijacking spam on Twitter.

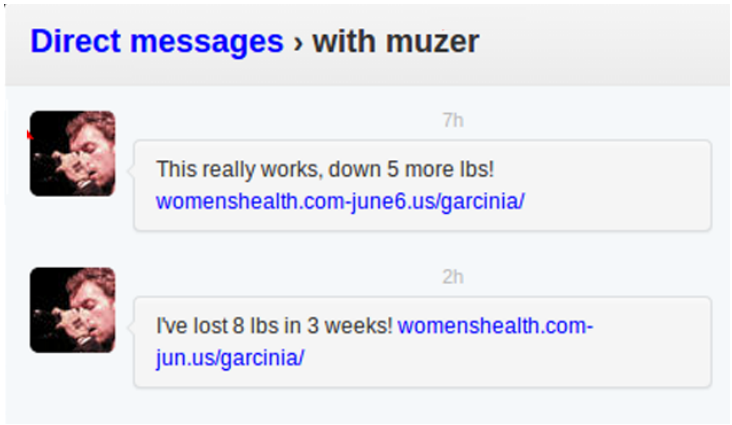

Figure 2.4: An example of direct messages spam on Twitter.

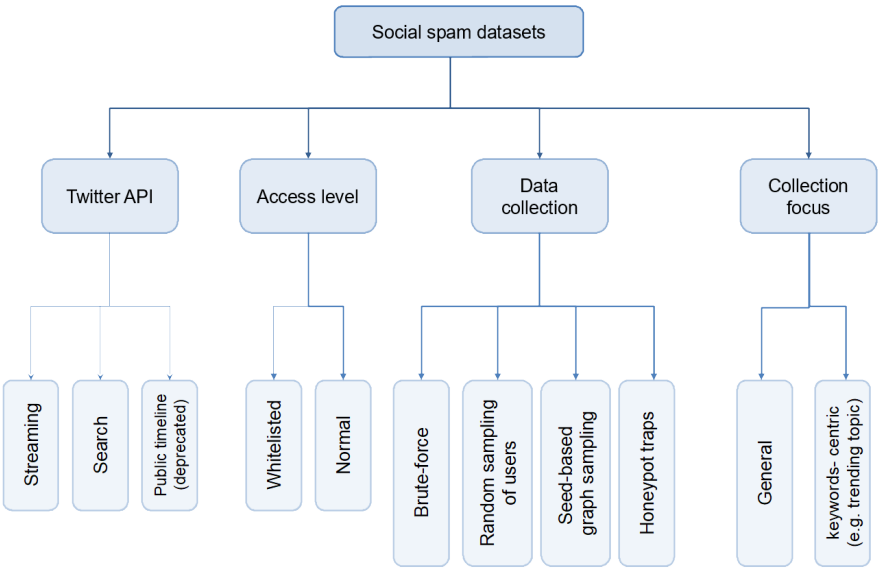

Figure 2.5: A taxonomy of social spam datasets on Twitter.

Chapter 3

Supervised Detection of Social Spammers on Twitter

Chapter Contents

3.1 Introduction . . . 43
3.2 Modeling Accounts as Numerical Features on Twitter . . . 45
3.2.1 General Profile Attributes . . . 46
3.2.2 Social Network Features . . . 46
3.2.3 Automation-related Features . . . 48
3.2.4 Behavioral Features . . . 48
3.2.5 Content-related Features . . . 49
3.3 Empirical Evaluation of Twitter's Suspension Mechanisms . . . 49
3.3.1 Compared Datasets . . . 50
3.3.2 Discussion . . . 52
3.3.3 Conclusion . . . 54
3.4 Detecting Spammers on Twitter: A Case Study . . . 54
3.4.1 Collection and Labeling of a Ground-truth Spam Dataset . . . 55
3.4.2 Evaluation of State-of-the-art Detection Features . . . 58
3.5 Conclusion . . . 64

3.1 Introduction

The supervised classification framework offers a straightforward way to model the components of the social spam detection problem. The solution design is typically restricted to mapping the detection objective (usually a single social account) to an appropriate set of statistical features. The problem can therefore be reduced to a task of features engineering where the goal is to propose and find the set of features that minimizes the empirical classification risk, for a given loss function. The loss function can itself be defined in several ways but the classification performance is usually

expressed in terms of traditional machine learning metrics (e.g. recall, precision, F1-measure). Evaluation in a supervised learning framework can therefore seem like a well-defined task. In practice, the performance of a classifier is closely tied to how the target is defined. The definition can vary significantly between different systems and studies. This is most clearly captured by how different a spammers population in a ground-truth dataset can diverge from Twitter's suspension mechanism as we will show in the rest of this chapter.

We have discussed in Chapter 2 a taxonomy of existing works that classifies a contribution according to two criteria: the detection objective (i.e., whether the detection system aims at detecting accounts, messages or URLs), and the detection granularity (contrasting individual detection of instances, such as one account, vs. community detection, e.g. detecting a community of accounts). Most of the literature contributions fall into the "individual account" detection category. The remaining chapters of the manuscript will model the classification task using this same detection objective. It is therefore important to thoroughly discuss the foundational contributions related to this modeling choice. Accordingly, this chapter has three goals. The first is to extensively review the features proposed in the literature to model individual social accounts. The second is to concretely discuss labeling and ground-truth dataset formation by comparing existing datasets. The third and final goal is to evaluate sets of features on a recent real-world Twitter dataset. This evaluation is undertaken to highlight the practical challenges of replicating state-of-the-art supervised detection systems and to motivate the need for a change in the mainstream supervised detection methodology.

In the studied case of spam accounts detection on OSNs, the supervised classification pipeline described above can be explicitly expressed as follows:

1. A social account is modeled as a numerical features vector $\boldsymbol{x}_i$ belonging to an input domain $\mathcal{X}$.

2. A supervised classification model is trained on pairs of features vectors $\boldsymbol{x}_i$ and their corresponding labels y_i. The classification task is usually binary and the label y_i is defined as a positive instance when the account is a spammer. Formally, $y_i \in \mathcal{L}$, where $\mathcal{L} = \{0, 1\}$. A spammer label is denoted by 1 while a legitimate label is denoted by 0.

3. The classification function $f(\boldsymbol{x})$ obtained by training the above-mentioned Machine Learning model maps the input domain $\mathcal{X}$ to the output domain $\mathcal{L}$.

Figure 3.1 shows a simplified supervised classification pipeline applied to spam accounts detection on OSNs. The three main components as outlined above are: account featurization into a vector $\boldsymbol{x}_i$, assigning a label y_i to the account (labeling), and training a classification model to obtain a classification function $f(\boldsymbol{x})$.

Chapter structure. The remaining of this chapter is structured as follows. Section 3.2 presents an extensive overview of the features used in the literature to model social accounts for the classification task. Section 3.3 illustrates the heterogeneous nature of labeling in Twitter's social spam research and empirically shows that Twitter's suspension mechanism is incomplete and suffers from delays. Section 3.4 details

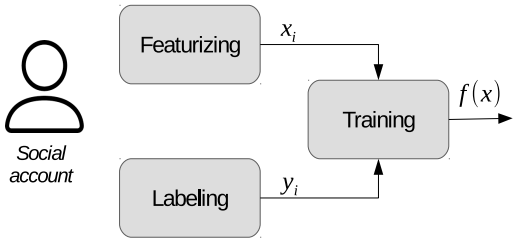

Figure 3.1: A simplified partial overview of the supervised classification pipeline as applied to spam accounts detection on OSNs.

training Machine Learning models on the sets of features derived from those discussed in Section 3.2. In doing so, the full training pipeline is exposed and practical challenges are discussed. The features evaluation is undertaken on a Twitter ground-truth dataset that we collect and annotate specifically for this task. We heavily base our discussion of data collection and labeling on the corresponding background section (Section 2.3). We evaluate and compare several of the main supervised classification models in the setup established by the features and the collected dataset. We use the results of the classification to discuss spam evolution and evasion and its impact on supervised classifiers. We also highlight specific conclusions that motivate the contributions we propose in the following chapters.

3.2 Modeling Accounts as Numerical Features on Twitter

In this section, we extensively review and classify features proposed in the literature to model social accounts. Since the focus is on individual accounts classification, we do not discuss features that can be extracted from a collection of accounts [80], nor features extracted from individual URLs [69, 85, 34, 86]. We also limit our discussion of features extracted from messages to those that can be directly exploited in accounts classification. We do not therefore discuss features that are defined for tweet level classification [81, 82, 83].

Figure 3.2 summarizes the areas over which features can be extracted to model a social account. These include:

- Profile attributes.
- Social Network attributes.
- Content attributes.
- Automation attributes.
- Behavioral attributes.

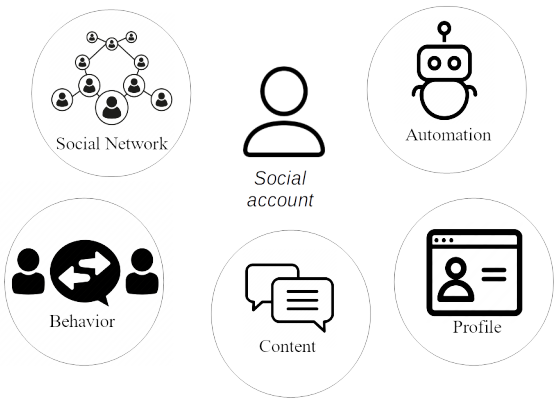

Figure 3.2: Different types of features extracted from a social account.

3.2.1 General Profile Attributes

We mean by general profile attributes the features that can be obtained by simply looking at the user profile, without applying any aggregations or temporal computation over the account activity or content. These features include the account's name, screen name and description, its age (or creation date), its total number of tweets and liked tweets as well as the number of lists containing the account. Since some of these characteristics are hard to analyze and interpret in their raw format, they are often used to compute simple statistics, such as the length of the name, screen name and description.

3.2.2 Social Network Features

Using features related to the social network of a profile is a legacy of the earlier work on social spam detection. The assumption was that spam accounts were peripheral accounts that had weak social ties and insignificant ego networks[1]. While this assumption was supported by empirical evidence [10, 11, 23], followers acquiring techniques that we discussed in chapter 2 quickly ensured that spambots followship improved significantly, and that simplistic features computed on an account's ego network were not effective at distinguishing genuine users from malicious ones [91]. These features include simple counts of the number of followers and friends of an account, as well as different ratios computing the relationship between these two counts:

- $\frac{friends}{followers}$
- $\frac{friends}{followers^2}$

[1] An ego network consists of a focal node, called *ego*, and all the nodes that connect to it (*alter*), along with all the edges that connect alter nodes.

Table 3.1: Advanced features computed over an account's ego network.

Similarity
1. Std. of followers IDs
2. Std. of friends IDs

Centrality & embeddedness
3. Clustering coefficient
4. Betweenness centrality
5. Reciprocity (bidirectional links ratio)

Quality & engagement
6. Avg. number of followers of neighbors
7. Avg. number of tweets of neighbors
8. Followings to median neighbors' followers
9. Number of followees of the user's followers
10. Number of tweets of the user's followees

- $\frac{friends}{friends+followers}$

Later work [13, 91] proposed new more complex features computed over a user's ego network (see the list of features in Table 3.1). The first two features shown in the table are proposed to measure a possible collusion between the accounts in the ego network. The assumption is that when these accounts have close numerical IDs, this implies that they have been created simultaneously and thus similar IDs can indicate a certain degree of collusion. Features $3-5$ measure the centrality of the studied account (betweenness) and how well embedded it is in its ego network using the clustering coefficient and reciprocity. The remaining features ($6-10$) attempt to quantify how real the account's neighbors appear to be. This is done by measuring their content and followship/friendship bases.

The authors hypothesize that each of these more complex features is more difficult to evade than the simplistic earlier features. The argument is that unlike buying followers, it is both harder and costlier to fabricate the proposed features (e.g. fabricate a higher betweenness). While this reasoning stands true for betweenness, it has become increasingly easy and cheap for fake followers to have high followers themselves (via mutual following), or to have active profiles (via automated posting scripts). A more recent investigation [76] of the power of these features in detecting fake followers has shown that the features as an ensemble are still very efficient. The problem however, as this same study notes, is that these features are very costly to compute as they require either processing over the whole ego network of each account (betweenness centrality), or aggregating numbers from each of its followers (average number of followers of neighbors). To these drawbacks, we add acquiring all neighbors (followers and friends of an account), and acquiring the connections that may exist between these accounts, which is required for constructing the full

Table 3.2: Features capturing the social and content-posting behavior of a social account.

Social Interaction (incoming and outgoing)
Following rate (out)
Befriending rate (in)
Mentions rate (out)
Reply rate (out)
Nb. of times the user was mentioned (in)
Nb. of times the user was mentioned per time period (in)
Nb. of times the user was replied to (in)
Nb. of times the user replied to someone (out)

Content
Reply ratio: proportion of replies
Retweets ratio
Original tweets ratio
Tweeting frequency
Distribution of tweets in temporal bins
Min, max, medium and mean of:
The time between tweets
Nb. of tweets posted per day/week

ego network of an account. The acquisition of this data is practically impossible, as this requires an inordinate amount of API calls for each account[2].

3.2.3 Automation-related Features

Since automation and spam accounts are strongly correlated, features quantifying the automation of an account are valuable in detecting spam accounts. Automation can be measured based on temporal posting behavior, content similarity, or the devices used to post content. We discuss the two first types in the following sections. Devices-related features measure how much of the content of an account is produced on the API interface rather than using the web interface or a mobile application (e.g. measuring the API tweets ratio, API URL ratio and API tweets similarity). This concept has admittedly not aged well, as the number of applications on Twitter is huge [32] and most of the automated content is posted through third party applications rather than using the official API portal.

3.2.4 Behavioral Features

Behavioral attributes of an account, listed in Table 3.2, are features that focus on measuring characteristics of the account activity. They can be broadly divided into

[2]Rates of API calls on the social graph of an account (e.g. the GET followers/ids request) are restricted to 15 calls per a 15 minutes time window.

features that measure the account's interaction with other accounts and features that quantify the account's posting behavior. The latter are not to be confused with content-related features, which we discuss in the next section.

Interaction features can be further divided into *incoming* features associated with the cases where other accounts initiate an action towards the studied account (by mentioning, replying or following), and *outgoing* features where the account itself initiates the action.

Content posting features attempt to profile the account's posting activity both temporally (tweeting frequency, inter-tweets time, temporal distribution) and based on the post type. While most of these features are expressed in terms of ratios, descriptive statistics can also be used for further differentiation between accounts.

3.2.5 Content-related Features

Content features are those features computed on the ensemble of tweets posted by an account. They can be roughly partitioned into the following categories. See Table 3.3 for a full list of features.

- Content-level features are features representing global statistics of the tweets, usually counts or frequencies of occurrence of special text elements or platform operators. These include for example the ratio and raw number of URLs.
- Aggregated tweet-level features are features issued from aggregated statistics of attributes computed individually on each tweet. While the global features offer a rough picture of the tweets pattern, these tweet-level features (e.g. number of words per tweet) offer a closer look on how individual tweets are created.
- Replication features are features measuring content variability. These are useful in detecting spam accounts, especially simplistic ones, as these tend to replicate the same content. Enhanced replications measures (using compression or similarity measures) can also help bypass the artificial variability introduced by spammers to avoid that their tweets be flagged as exact replicates.

3.3 Empirical Evaluation of Twitter's Suspension Mechanisms

Twitter has an extensive and detailed set of rules that outline what the platform considers as abuse [109]. It does not, however, disclose the internal workings of the monitoring, detection and suspension systems. We do know that reporting constitutes an important part of the defense ecosystem, and that the platform has been recently alleviating its dependency on manual reporting and review [109]. Unlike the glimpse offered by the article on Facebook's so-called *immune system* [110], however, we do not have any inside information on the detection algorithms used by Twitter.

The only visible part of Twitter's defense mechanisms is suspension, a strong signal that constitutes, as we have previously underlined, a business decision that may induce real world implications to the company's public image and economical

Table 3.3: Features computed on an account's content (collection of tweets).

Content-level
URL ratio: Proportion of tweets containing a URL
Mention ratio
Hashtag ratio
Nb. URLs
Nb. Hashtags
Nb. Tweets with @ (reply/mention)
Nb. Mentions
Nb. URLs per day
Nb. hashtags per day
Nb. mentions per day
Ratio of tweets containing words from a popular list of spam words
fraction of tweets with a blacklisted URL

Aggregation of tweet-level attributes
Min, max, medium and mean of (computed over each tweet):
Nb. of hashtags per number of words
Nb. of URLs per words
Nb. of mentions per words
Nb. of words
Nb. of characters
Nb. of URLs
Nb. of hashtags
Nb. of users mentioned
Nb. of times the tweet has been retweeted
Nb. of times the tweet has been favorited
Nb. of numeric characters

Replication
Avg. similarity over all pairs of tweets
ZIP compression ratio of tweets
Nb. of replicates

worth. We have previously stated in Section 2.3 that the suspension process is incomplete, suffers from delays and does not neatly map to a pre-defined definition of abuse or spam. In this section, we support these claims by measuring and comparing the suspension rates of accounts in 7 different Twitter datasets. The discussion in this section focuses only on the heterogeneous nature of ground-truth datasets with respect to the definition of the objective, i.e., the social spamming account. Another heterogeneity factor stems from the different formats of these datasets, which will hinder replication as we will discuss further in Section 3.4.

3.3.1 Compared Datasets

The compared datasets (presented in Table 3.4 below) cover the major collection and labeling methods (as discussed in Section 2.3) and span a decade of research between

2009 and 2018. They offer an excellent benchmark for measuring and exploring suspension on Twitter.

3.3.1.1 Checking for users suspension

All of the seven datasets contain numerical identifiers (IDs) of Twitter accounts. We have checked the state of each account by connecting to the users lookup interface of Twitter's API. The returned response offers coarse information indicating whether the page associated with an account ID exists. For the IDs that do not have an associated page, we have conducted a second round of verification to determine if the account has been suspended by Twitter or simply deleted. This is done by individually crawling each of the non-reachable users identifiers. The results are shown in Table 3.4 under deletion and suspension percentages respectively.

3.3.1.2 Accounts labeling

Although the datasets are very different in terms of how accounts are collected and labeled, and how abusive behavior is defined, we can nonetheless divide the accounts into three large categories: spam accounts (showcased in red in Table 3.4), legitimate and human accounts (showcased in green) and randomly sampled accounts (showcased in blue).

Each of these datasets has been built in the context of spam detection on Twitter. There exist two broad approaches to balance the spam population and sample non-spamming accounts. The first is random sampling of the platform's accounts and the second is collecting genuine or alternatively "human" accounts. Note that this distinction is not clearly made in the respective literature as random accounts are generally used interchangeably with legitimate accounts. We make this distinction here as it allows to better compare the difference between manually annotated legitimate accounts and the general Twitter population. This further allows to examine the common assumption that spammers form a negligible percentage of the general population.

Spam accounts. The spam accounts are obtained either by identifying blacklisted URLs (the 1k-10k dataset), by honeypots (the honeypot dataset), by manual review of suspicious hashtag-hijacking content (the hashtag-hijackers dataset), by searching for common catch phrases used by Tinder-marketing bots (the pr0n dataset), by buying accounts from fake followers services (the Cresci-2015 dataset), or by a combination of manual review and search for affiliate marketing accounts (the Cresci-2017 dataset).

Randomly sampled accounts. Three datasets contain a random sample of the Twitter's sphere, the earliest one being the honeypot dataset that was collected in the period between 2009 and 2010. The 1k-10k dataset also offers a random sample of accounts. The authors choose to collect the dataset by means of Breadth-First graph search, which offers a less random sample than the one offered in the honeypot dataset. We have also conducted a sampling of the population by sampling tweets

from the Twitter streaming API in April 2018, and constructed a dataset containing $118,093$ distinct accounts (the streaming API sample dataset).

Legitimate accounts. Compared to random accounts, properly legitimate accounts are generally obtained with a process that requires a more pronounced human involvement. This translates in practice to the accounts being manually labeled as genuine (e.g. in the case of the hastag-hijackers and Cresci's E13 datasets) or alternatively being verified by a human after a community collection process. In the case of the TFP dataset (Cresci-2015), the users were collected via an academic initiative called the fake project, which aimed at collecting genuine accounts. The genuine accounts in Cresci-2017 were obtained by randomly contacting accounts on Twitter, and keeping accounts that were able to answer a question (a sort of a social Turing test).

3.3.2 Discussion

The obtained suspension rates are shown in Table 3.4. Datasets are sorted in an ascending order according to their collection rates. The table also shows suspension and deletion rates of each dataset and states the dates in which the suspension checking was done. Since the used datasets do not offer a true distribution of spammers (resp. legitimate users), it is not useful to compute and compare aggregated average suspension rates over the different datasets. The numbers can serve to show that Twitter's suspension system does indeed suspend more spammers on average than users labeled as genuine. Similarly, users labeled as genuine are less likely on average to be suspended than randomly selected users.

Percentage of spammers in the general population. The suspension rates in random samples in Table 3.4 range between 5% and 9.5%. This seems to back the estimates discussed in Section 2.3.2.3. It should be noted that, due to suspension delay and incompleteness, the reported suspension rates represent lower bounds on the true spammers ratio in the general population. Suspension delay and incompleteness are discussed below.

Suspension delays. To assess the time delay of the suspension mechanism, we evaluate the suspension rates on the randomly sampled "Streaming API" dataset on two different dates. The first assessment is conducted in July 2018 (4 months after the dataset is collected) and the second is conducted in April 2020, two years later. The difference between the rates reported in Table 3.5 confirms the delay in the suspension process (more than $4,000$ additional accounts were suspended and more than $12,000$ additional accounts were deleted).

Suspended vs spam accounts. There is an extreme variation in the suspension rates of spam populations reported in Table 3.4.

Table 3.4: Suspension and deletion rates of accounts in seven Twitter datasets. Spammers samples are shown in red, legitimate samples are shown in green, and random samples are shown in blue.

	Dataset	Accounts type	Accounts	Percentage deleted	Percentage suspended	Date collected	Date checked
1	1k-10k	Spammers (blacklisted URLs)	1,000	15.8%	12.7%	<2011	1/2020
		Graph sampled - BFS (from random seeds)	10,000	18%	9.5%		
2	Honeypot	Identified by honeypots	22,223	28.8%	22%	12/2009 – 08/2010	3/2017
		Random sample	19,276	16.4%	0.7%		
3	hashtag_hijackers	Hashtag hijackers	165	66.7%	66.7%	11–12/2014	3/2017
		Legitimate (manual)	165	5.5%	3.0%		
4	Pr0n bots	Explicit marketing bots	80,157	99.5%	NA	3/2018	4/2020
5	Streaming API sample	Random sample	118,093	13.4%	5.2%	4/2018	7/2018
6	Cresci-2015	Humans (TFP)	1,481	14.1%	0.2%	12/2012	1/2020
		Humans (E13)	469	7.0%	0.2%	2013–2015	
		Fake followers (FSF)	1,169	97.2%	97.2%	4/2013	
		Fake followers (INT)	1,337	94.9%	72.3%	4/2013	
		Fake followers (TWT)	845	15.3%	9.3%	4/2013	
7	Cresci-2017	Genuine accounts	3,474	14.7%	1.5%	<2014	1/2020
		Social spambots 1	991	49.1%	17.9%	2014	
		Social spambots 2	3,457	4%	3.9%	<2014	
		Social spambots 3	464	18.1%	17.0%	<2014	
		Traditional spambots 2	100	99%	99%	<2014	
		Traditional spambots 3	403	50.4%	50.4%	<2014	
		Traditional spambots 4	1,128	31.7%	31.7%	<2014	

Table 3.5: Evolution of suspension rates in the "Streaming API sample" dataset between July 2018 and April 2020.

Date checked	Percentage deleted	Percentage suspended
7/2018	3.2%	1.5%
4/2020	13.4%	5.2%

- Some categories of spammers seem to be easily identified by Twitter's suspension system (with rates in the range of 94%-99% for the *pr0n bots*, the *traditional spambots 2* dataset and the *fake followers* from the *FSF* and *INT* datasets).
- Spammers in other datasets seem to be more difficult to identify. The spammers in the two legacy *honeypot* and *1k-10k* datasets in particular, have surprisingly low rates of 22% and 12.7% despite being identified a decade ago.

An interesting observation is that fake followers from different campaigns bought at the same time period equally differ in terms of suspension rates. The vast majority of accounts in the FSF and INT campaigns have been suspended. But only 15% of the accounts bought from the TWT campaign have been suspended.

3.3.3 Conclusion

The discussion above shows that there is no direct correspondence between Twitter's suspension requirements and the definition of spam accounts in the different discussed works. The discussion also confirms that Twitter's suspension process is incomplete, i.e., that it does not suspend all accounts that violate the platforms rules of service. Moreover, the suspension rate varies significantly from one dataset to another. Simplistic accounts and fake followers seem to be easily and consistently detected by Twitter's suspension mechanisms while more complex accounts are harder to detect.

3.4 Detecting Spammers on Twitter: A Case Study

The goal of this section is to evaluate sets of features proposed in the literature and reflect on the implications of the results and classifiers performance.

An ideal evaluation would require exact *replication* of the detection systems described in the literature. This would require extracting the same features used in the proposed systems, and an access to the dataset on which the Machine Learning (ML) models were trained. Exact replication would require, moreover, not only the ML model used to train the classifier (e.g. Decision Trees, Naive Bayes, ...), but also the exact experimental parameters used for training (e.g. the train/test data split, the initial training conditions). While papers usually describe the ML model they use, sharing experimental parameters is regretfully not a common practice. As for datasets, when a dataset is shared, it often has a restricted format that hinders its use

for replicating the corresponding classification system. This point will be discussed in more details in Section 3.4.1 where we also describe a custom dataset that we collect and label specifically for the task of evaluating supervised detection systems. Given that the only reliably divulged component of these systems is the set of features on which they are based, we only use sets of features to replicate and evaluate these systems. Note that works such as [77, 14] that similarly evaluated state-of-the-art systems relied only on the described sets of features to replicate the systems for the same reasons we discussed above.

Evaluating the performance and efficiency of a set of statistical features requires an experimental setting where a ground-truth dataset is defined and a classification model as well as a loss function are chosen to train the classification model. Like many domains where the input distribution is not stationary (e.g. credit card fraud, email spam), it is valid to have questions on the performance of state-of-the-art social spam classifiers when experimental settings change. Note that evidence of a drift in the input distribution has been noted as early as 2011 [13], while recent reports have also underlined the underperformance of some state-of-the-art classifiers on recent datasets, coining the phenomenon as a "paradigm change" [14].

3.4.1 Collection and Labeling of a Ground-truth Spam Dataset

A number of datasets containing annotated Twitter accounts has been introduced over the years (e.g. in [10, 13, 14] and on the bot repository[3]). With the exception of a couple of early datasets [13, 10], the shared files corresponding to these datasets often consist of users identifiers and eventually some of their messages identifiers (see an example from the bot repository in Table 3.6). This can be traced back to Twitter's terms of service, which state that researchers are not allowed to share users' content of their datasets externally[4]. This restriction means that replicating any work based on these datasets would require crawling the pages associated with the numerical identifiers. Since Twitter executes regular purges of suspicious accounts, content and profiles of suspended and deleted accounts in the aforementioned datasets is often inaccessible[5]. This makes results replication and exploitation of the labeled users infeasible.

Exceptions to the above sharing format include:

- The dataset of Benevenuto et al. [10], that provides a features matrix where each user is assigned a numerical vector associated with the features defined in the original work [10], but does not provide users' identifiers.

- The dataset by Yang et al., used in [13, 91, 112] where some users messages are provided in a text form.

[3]The Bot Repository `https://botometer.iuni.iu.edu/bot-repository/datasets.html`.

[4]Twitter's policy on research use cases `https://twittercommunity.com/t/policy-update-clarification-research-use-cases/87566`

[5]See [111] for unique insights on the volatile nature of manipulative content on Twitter in the context of Brexit online discussions.

Table 3.6: A snippet from the *vendor-purchased-2019* dataset on the bot repository. This illustrates the format of publicly available Twitter datasets.

user_id	label
736629545512632320	bot
764765673172176897	bot
830692986459738114	bot
872525677484007427	bot
795374336823721988	bot

Table 3.7: Characteristics of the ground-truth dataset

Group Designation	Class	Users	Tweets
Verified Users	Legitimate	500	100 108
Human Users	Legitimate	130	56 663
Trends Hijackers	Spammer	51	22 586
Promotional Spambots	Spammer	86	31 404
Total		767	210 761

- The dataset by Cresci et al. [14], that provides users features and text messages.

While these datasets differ from the bulk of existing datasets in that the original work itself can be replicated in [14, 10], an important drawback to note is that features issued from other works cannot be usually computed or evaluated on these datasets.

Since our goal is to replicate previously proposed features and compare and contrast their performance on the same dataset, we had to recur to custom collection and labeling of a ground-truth Twitter dataset[6]. The dataset contains 767 users[7] divided over four categories of users: verified accounts, normal users, hashtag hijackers and promoters. The first two categories belong to legitimate accounts and form 83% of the dataset, while the other two categories form the remaining 17% and exhibit an abusive behavior that violates Twitter terms of service[8]. Table 3.7 summarizes the general characteristics of the ground-truth dataset. For each of these users, Twitter's *Rest API*[9] was used to crawl users profiles and tweets. These were subsequently used to extract relevant content and behavioral features.

We explain hereafter the techniques used to collect and label Twitter accounts. Note that to obtain some users (e.g. users in the Verified category as well as some human users and promotional spambots), we needed to first collect a large dataset of random accounts and tweets. For this, we used the Developer *Streaming API*[9] in the

[6]The dataset (users ids, features and users graph) is available via `https://nourmawass.wordpress.com/datasets/`.

[7]The number of users in our dataset is comparable to other datasets obtained via manual labeling e.g. 759 users in RTbust [60], 62 and 529 accounts in [61] and 1065 accounts in [10].

[8]Twitter terms of service `https://twitter.com/en/tos`

[9]Twitter developers API `https://developer.twitter.com/en/docs`

Figure 3.3: A screenshot of a compromised verified account posting a tweet containing a phishing link.

period between 5 and 21 October 2017 and obtained a random sample of $20M$ tweets from $12M$ active users. For the remaining accounts in the ground-truth dataset, the collection targeted trending hashtags, the content of which was collected via the *Search API*.

Verified Accounts. Given the complex nature of accounts automation on Twitter, we found it important that the dataset comprised automated users on both ends of the spectrum, that is, automated profiles from both legitimate and abusive users. Verified users often belong to companies, celebrities or public figures, and are often operated by dedicated or generic content management applications[10]. They exhibit a behavior typical of what has come to be known in the literature as a "cyborg" account. These accounts may therefore have different features from those of normal human-based accounts and it is important to include them in the dataset to prevent the classifier from learning that every automated behavior is abusive.

these users are easy to identify (their profiles are marked with a blue tick mark and their crawled profiles include a "verified" flag). We randomly selected 500 users among 43k verified users appearing in the dataset and we included these 500 users in the ground-truth dataset.

Human users. The remaining 134 legitimate users in the ground-truth dataset were normal human-operated accounts. These users were identified by manually investigating a sample of active accounts from the initial dataset. This required a careful examination of the account in question, its tweets, profile and behavioral characteristics, and has therefore a small throughput. Manual labeling is different from mainstream labeling techniques described in the literature in that it is time consuming and requires an annotator that is familiar with current spam techniques and tricks[11].

Promoters. The blacklisted links heuristic is a well-known heuristic that is commonly used to identify spammers in email and social media [86, 69]. It consists of

[10]Examples of generic content management applications include *TweetDeck* and *dlvr*.

[11]Previous work that uses manual labeling such as [10] relies on crowdsourced annotation of individual hashtag tweets. While we think that this method could have yielded trustworthy annotation back when spam was less complicated and more straightforward, recent empirical evidence [113, 14] suggests that non-initiated human annotators fail to identify the new generation of spam on social media.

identifying users that post links to malicious webpages by verifying links appearing on social media against a continuously updated database of malicious webpages such as Google Safe Browsing[12] and Phishtank[13].

We applied this heuristic to the crawled dataset. For this, we first started by extracting all $3.8M$ links in the $20M$ crawled tweets. We subsequently wrote a program that follows the redirection chain of each link and returns the final landing webpage[14]. We then used Google Safe Browsing API to identify suspicious URLs. Only 156 URLs were identified as being phishing or malware URLs. We extracted all users IDs that posted any of these malicious URLs and then proceeded to the manual verification of the resulting accounts. Surprisingly, a significant number of these accounts were actually legitimate accounts that were temporarily compromised[15] by a malicious posting mechanism[16]. Consequently, we could not rely on this labeling heuristic alone to obtain malicious accounts as it yielded a high false negative rate. Alternatively, for the users that were found to be genuinely malicious, we extracted the text associated with the blacklisted URLs. We then searched Twitter for users that posted the same text, and were able to identify several communities of spammers. We obtained 86 users in total, most of them engaged in promotional and site referral activity.

Trends hijackers. Trend hijacking is a type of collective-attention spam [55] that is particularly ubiquitous on social media. It consists of poisoning trending topics (which typically offer high visibility and attract a large audience) with unrelated posts, often to promote a particular product or service (see Figure 2.3 for an example) or to manipulate public opinion [6, 7].

We obtained 47 trends hijackers by reading the tweets of a trending sport-related hashtag and manually identifying suspect tweets. This was followed by a manual investigation consisting of reading the recent tweets of suspect profiles and cross-examining different profiles for similar patterns and content. This process is similar to the one described in [114, 73].

3.4.2 Evaluation of State-of-the-art Detection Features

3.4.2.1 Statistical Account Features

As our goal is to base users prior predictions on established methods from the literature, we limited the analysis to these methods and did not endeavor to propose new features to model social accounts.

[12]Google Safe Browsing API: `https://developers.google.com/safe-browsing/`

[13]The Phishtank database `https://www.phishtank.com/`

[14]To detect dynamic redirection, we used the *selenium* Python package to open each URL in a browser window.

[15]Compromise is fairly common on social media. We used a variation of the Compa system described in [37] for identifying and excluding compromised accounts among identified suspicious accounts. Compa builds statistical profiles for users and identifies compromise by comparing recent posts with the previously built profile.

[16]In one instance of these compromise campaigns, the "Rayban sale" scam, one verified account was found to retweet the same malicious URL dozens of times before the malicious behavior stops and the account restarts its normal behavior (see Figure 3.3).

Table 3.8: Description of features used in this work

	Feature	Description	Our features	Benevenuto	Stringhini
Profile	age of the account	time since the account was created	✓	✓	
	statuses count	Total number of tweets posted by the account	✓		✓
S. Network	followers count	Number of users following the account	✓	✓	
	friends count	Number of users followed by the account	✓	✓	✓
	followers per followees	Ratio of followers to followees (friends)	✓	✓	
	followees per squared followers	Ratio of friends per the squared number of followers			✓
Content	replicates	Number of identical posts in the account timeline	✓		
	similarity	Average similarity between the account's posts	✓		✓
	fraction of tweets with urls	Fraction of tweets containing a link	✓	✓	✓
	fraction of tweets with hashtags	Fraction of tweets containing a hashtag	✓		
	fraction of replies	Fraction of replies in the user's posts	✓	✓	
	fraction of retweets	Fraction of retweets in the user's posts	✓		
	mean nb hashtags per tweet	Average number of hashtags in a tweet	✓	✓	
	mean nb urls per tweet	Average number of links in a tweet	✓	✓	
	urls used on average	Average number of times a single link is used	✓		
Behavioral	avg intertweet interval	Average time interval between consecutive tweets	✓		
	std intertweet interval	Standard deviation of time intervals between consecutive tweets	✓		
	min intertweet interval	Minimum time interval between two consecutive tweets	✓		
	nb followees per day	Average number of followees the account follows in a day	✓		
	nb followers per day	Average number of users following the account in a day	✓		
	active tweeting frequency per day	Average number of tweets posted by the account on a daily basis (computed based on the 200 most recent tweets)	✓		
	distribution of tweets in temporal bins	8 features corresponding to the proportion of the account's tweets contained in temporal bins of 3 hours each.	✓		

We select 28 features from different previous works [10, 11, 12, 63] and compute their values for accounts in the dataset. A list of these features along with their description is presented in Table 3.8. This set captures a wide range of information including aspects related to the accounts behavior, social network, content and social profile. We also specifically reproduce the works in [10] and [11] (denoted hereafter as *Benevenuto* and *Stringhini* respectively) which represent subsets of the larger set of features. These were chosen based on self-reported performance, wide acceptance in the community, and reproducibility. The latter is defined by the possibility of reproducing the model with accessible account information and without the need for internal information such as IP addresses or the social graph[17].

A main assumption of the proposed solution is that suboptimal systems (particularly supervised classifiers issued from the literature) can be used to reliably discover seeds of spam accounts.

The studies we have used to establish the prior predictions baseline offer a comprehensive set that is still used to evaluate supervised features-based systems. For example, Cresci et al. [77] evaluate features-based detection using features from Stringhini et al. [11] and Yang et al. [13] for the task of detecting *fake followers* on Twitter. The authors state that features proposed in other studies can be "assimilated" to features proposed in these two papers. In other words, these features sets can be used to represent other proposed features sets.

In the accounts featurization phase of the work, we have extracted a large list of 145 attributes from each account. See the full list of attributes in Appendix A. These attributes cover most of the features proposed in the supervised literature, as well as features we have proposed ourselves in an earlier work [73]. Importantly, these attributes are extracted from the *API* data. This means that the extracted list is even more representative of accounts than some of the more recent literature that aims at detecting automated Sybils on Twitter. The work in [78] for instance is based solely on attributes extracted from the Document Object Model (DOM) content of the retrieved Twitter accounts in the browser. It does not exploit API-based features and therefore offers a more limited representation of a social account. For these reasons, the features used to represent Twitter's accounts in this chapter, despite being based on earlier literature, are more relevant to the accounts classification task than those that were proposed or used in more recent literature.

Figure 3.4 shows cumulative distribution functions (CDFs) of the top 9 individually relevant features selected by the mutual information method. The curves distinguish three types of users: verified, humans and spammers. They confirm that verified users are indeed different from humans and sometimes exhibit behavior closer to spammers. Some of the cumulative distributions associated with spammers have more than one inflection point. This suggests that the spammers population has a dual distribution with respect to these features (e.g. proportion of retweets, ratio of tweets containing URLs). This empirical observation generally supports the proposition that spammers are not a homogeneous population. The distribution of some

[17] While it is certainly possible to use Twitter's Rest API to obtain a user's social graph, the imposed API rate limit makes it prohibitive and impractical to require this information in a large-scale model. Models using such information (e.g. [13]) are hard to reproduce with a normal-level API access.

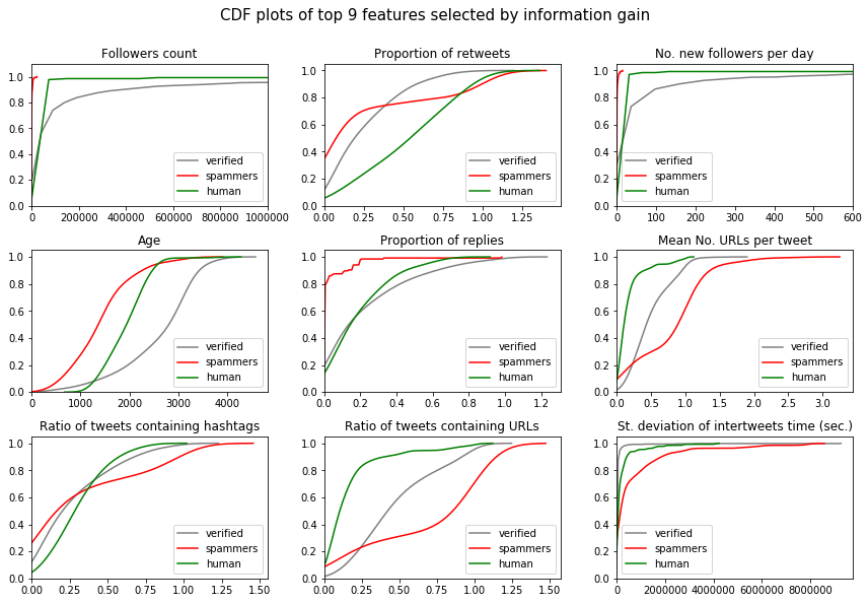

Figure 3.4: The CDF plots of the top 9 relevant features in the ground-truth dataset as selected by the mutual information method.

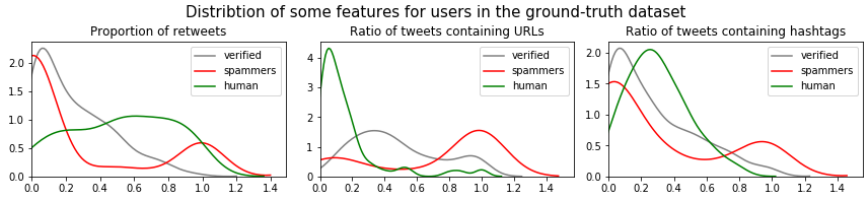

Figure 3.5: Distribution plots of features showcasing a dual spammer behavior.

of these features are shown in Figure 3.5. These distributions show that spammers' behavior can closely mimic that of verified or human users.

3.4.2.2 Choosing Machine Learning Classification Models

We train and evaluate the described set of features using the following discriminative models: Support Vector Machines (SVM) [115], Logistic Regression (LR) [116] and Random Forests (RF) [117]. We do not to use generative learning models (e.g. Naive Bayes) to avoid learning a joint distribution $p(x, y)$ over the input and output spaces. This is done because the collection of Twitter ground-truth datasets introduces a selection bias and the resulting dataset does not offer a true distribution $p(x)$ over the input space. We only want therefore to learn the conditional probability $p(y|x)$. Note that, although deep neural networks can also be used as a local supervised

Table 3.9: The classification confusion matrix.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive TP	False Positive FP
	Negative	False Negative FN	True Negative TN

classifier, the limited size of the available datasets does not allow an effective deep implementation.

3.4.2.3 Evaluation on the Ground-truth Dataset

We use the *scikit-learn* library [118] in Python to train the SVM, Logistic Regression (LR) and Random Forests (RF) classifiers. The SVM classifier uses the RBF kernel, and its parameters C and γ are obtained using a grid search. For LR, we compare results with $L1$ and $L2$ regularization. We evaluate and compare the classifiers over the three previously discussed sets of features, namely our selected set of state-of-the-art features and the sets of features proposed in Benevenuto and Stringhini. All features are normalized before training.

Given the unbalanced nature of the classification problem and the ground-truth dataset, we only consider precision, recall and the F1-measure for performance evaluation, as focusing on accuracy is misleading. These measures are defined with respect to the confusion matrix (Table 3.9) as follows:

- $Precision = TP/(TP + FP)$.
- $Recall = TP/(TP + FN)$.
- $\textit{F1-measure} = 2 \times \textit{Precision} \times \textit{Recall}/(\textit{Precision} + \textit{Recall})$.

Table 3.10 shows the average classification results for a 5-fold cross validation repeated 100 times. Although the compared models are better than either a random or a majority classifier[18], the classification results are clearly not in the range expected of a production-level system. In all instances, the reported performance measures are significantly lower than those reported in the original papers. Stringhini et al. [11] report a precision of 97% and a recall of 97.4%, while Benevenuto et al. [10] report a recall of 70% and a precision of 90.5%. For Stringhini features, there is also a particularly pronounced gap between precision and recall.

Note that since both training and evaluation were undertaken on the same dataset, the deterioration in performance cannot be explained by the difference between the original data distribution (e.g. in Benevenuto) and that in our dataset.

[18]Since this is an unbalanced classification problem, it is interesting to compare the supervised classifiers performance to the performance of a random classifier. Since 80% of the population is comprised of legitimate accounts, we define a random classifier that assigns labels randomly with a probability of 80% for the legitimate label. This results in a precision and recall of 0.2 and an $F1$ of 0.05, which is outperformed by the supervised classifiers in Table 3.10.

Table 3.10: Average classification performance of supervised classifiers on the ground-truth dataset.

		Our features	Benevenuto features	Stringhini features
SVM	Precision	0.454	0.686	0.777
	Recall	0.457	0.262	0.121
	F1	0.455	0.347	0.208
LR L1	Precision	0.593	0.755	0.796
	Recall	0.582	0.705	0.109
	F1	0.586	0.729	0.184
LR L2	Precision	0.527	0.671	0.872
	Recall	0.5	0.576	0.068
	F1	0.512	0.619	0.124
RF	Precision	0.532	0.807	0.842
	Recall	0.492	0.763	0.188
	F1	0.509	0.784	0.295

It can be instead argued that the deterioration in performance is due to a substantial part of spammers changing their behavior and characteristics, thus affecting the predictive power of the features proposed in the literature and driving the performance down. This is most pronounced when evaluating the Stringhini features [11].

From prediction to probability. The series of plots in Figures 3.6, 3.7 and 3.8 represent the distribution of the probability of the spam class as predicted by the classifiers trained over the whole labeled dataset. For all classifiers except the random forest (an ensemble classifier), the distribution of the probability predicted by the classifier has a bimodal distribution. The positions of the modes and the form of the distribution vary depending on the classifier and the features set. The important observation remains that one mode is positioned above the 0.5 threshold ($p > 0.5$) and the other positioned below the threshold. This seems to support the hypothesis that some spammers are still detectable (distribution centered in the positive prediction area) while others have evolved.

The features also seem to play an important role in the form of the distribution. The distribution corresponding to the Stringhini set of features (Figure 3.8), for example, has its minor mode in the positive prediction region, while the opposite can be said for the distribution corresponding to our selected set of features, where the major mode resides in the positive prediction region.

Conclusion. Training classifiers over different sets of features results in a performance that is considerably lower than was originally reported in state-of-the-art systems. Moreover, the classification performance varies significantly from a features set to another but is more consistent across different classifiers on the same features set.

The major hypothesis explaining these results is that spammers have changed their characteristics, in what is usually coined as spam evolution. The probability predicted by each trained classifier follows a distinct bimodal distribution which can be interpreted with a rough categorization of spammers into two sub-populations. Following this logic, the detected spam accounts are those that still follow patterns identified by previous works, and are thus detectable by previously proposed features. The opposite is true for spammers that have been misclassified by the trained classifiers.

The results validate the need to explore alternative approaches to traditional supervised classification. In particular, it can be conjectured that these classifiers can be used for reliable *discovery* of seed spam accounts, where seeding is defined as the stage where a collection of accounts is used to initialize a detection system. Moreover, using the prediction probability instead of the binary predicted class can serve as a belief in a probabilistic framework. The probability distributions do indeed indicate that legitimate accounts are concentrated around $p = 0$ whereas spammers have a wider prediction range. These propositions will be more amply discussed in the next chapter.

3.5 Conclusion

Supervised learning is a powerful paradigm that has been extensively exploited to create defense systems against social spam. Features engineering represents a central part of what defines supervised detection systems. The definition of the detection objective represents another important part.

In this chapter, we focused on features-based supervised classifiers and considered the spam account as the detection objective. We denoted the reviewed supervised systems as features-based to clearly distinguish them from systems based on probabilistic graphical models, which we propose in the next chapters. These proposed systems combine both accounts features and a graph over accounts.

We presented an extensive overview of the features used in the literature to model social accounts (Section 3.2) and compared the spam definitions that different datasets used and the impact of the ground-truth data collection on accounts suspension on Twitter (Section 3.3). These discussions underlined the heterogeneity of supervised features-based detection systems in terms of accounts modelling and spam definition. The discussion also highlighted the challenges in replicating and cross-comparing different studies.

In the absence of an acceptable benchmark, we collected and labeled a ground-truth Twitter dataset, using features from different state-of-the-art systems to model accounts (Section 3.4). The analysis we performed on the classification results has two main outcomes:

- Features used in the literature to model accounts cannot be used to reliably identify spammers in a population of social accounts. This may suggest an evolution of spam accounts, possibly with the goal of evading detection, that lead to a deterioration in the discrimination capacity of the evaluated features.

- The class probability predicted by the trained classifiers offers a better signal than the binary predicted class. The probability distribution of the spam class is notably multimodal, which may suggest an uneven evolution in the characteristics of spam accounts.

We used these two observations to motivate a shift in the usage of features-based classifiers. Instead of *detection*, we propose to use them in the following chapters in a *seeding* and *discovery* phase. We argue in the next chapter that the class probability predicted by these classifiers can be effectively exploited in the context of a probabilistic model.

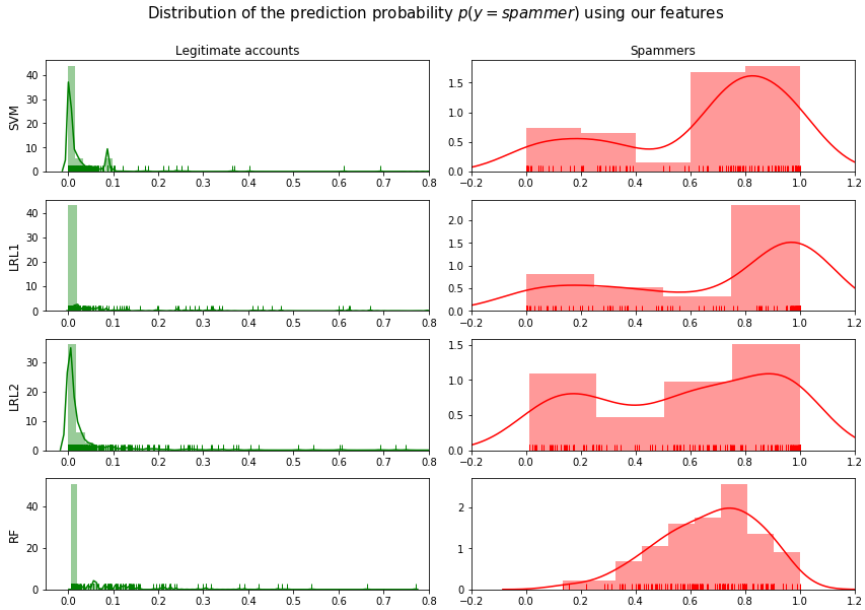

Figure 3.6: Distribution of probabilities computed by multiple classification models on the accounts represented by our set of features.

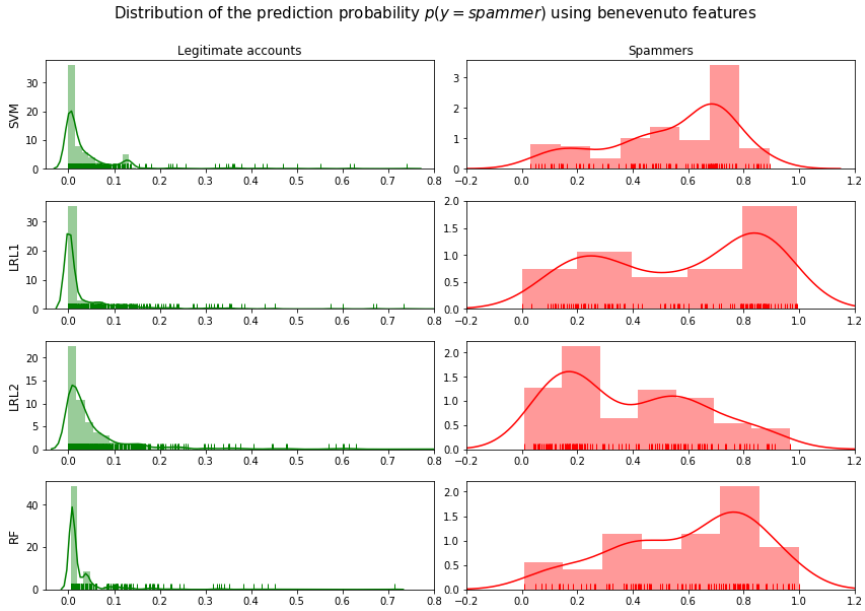

Figure 3.7: Distribution of probabilities computed by multiple classification models on the accounts represented by Benevenuto's set of features.

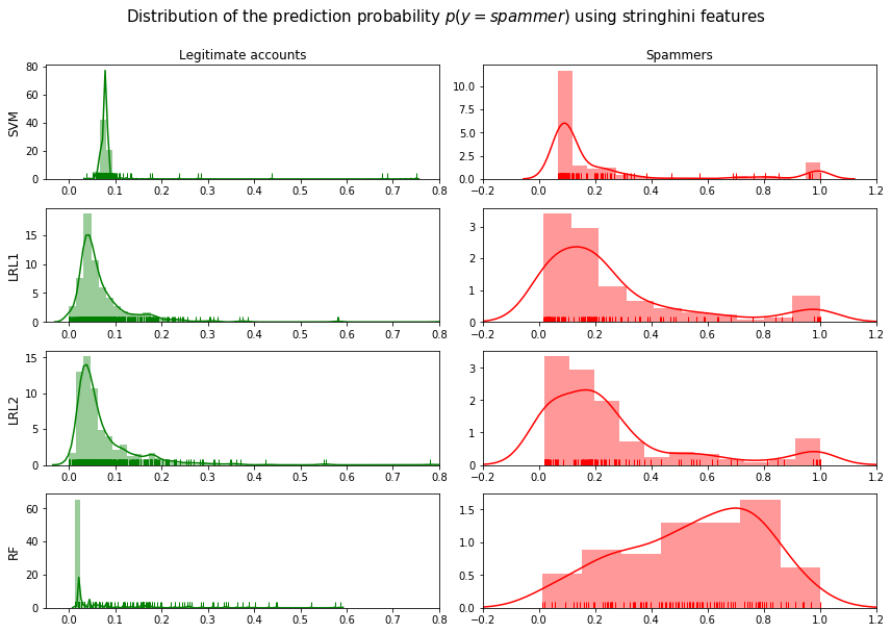

Figure 3.8: Distribution of probabilities computed by multiple classification models on the accounts represented by Stringhini's set of features.

Chapter 4

Enhanced Social Spammers Detection on Twitter using Markov Random Fields

Chapter Contents

4.1 Introduction 69
4.2 Probabilistic Modeling of Spam Classification 71
4.2.1 An Informal Introduction to Probabilistic Graphical Models . 71
4.2.2 Formal Definition of the Spam Classification Task 73
4.3 Proposed MRF Solution 73
4.3.1 General Solution Design 74
4.3.2 Markov Random Field for Social Spammers Detection 75
4.4 Constructing a Users-similarity Graph 78
4.4.1 Representing Similarity on OSNs 79
4.4.2 Proposed Content-based Similarity 80
4.5 Experimental Evaluation and Discussion 88
4.5.1 Experimental Setting 89
4.5.2 MRF Classification Results 90
4.5.3 Discussion and Generalization Insights 93
4.6 Conclusion 94

4.1 Introduction

We have established in the previous chapters that the volume and velocity of content generation on OSNs, make it imperative to recur to filtering spam via automated ways. We have also established that most of the contributions in the social spam community adopt the supervised detection paradigm. This is due to the relative simplicity and accessibility of the model.

In the conclusion of the last chapter, we have touched upon concept drift, a topic that is very relevant to real world machine learning applications in general and to spam detection in particular. By evaluating state-of-the-art detection features on a recent Twitter dataset, we have empirically demonstrated a disintegration in the performance of these previously successful systems, a result that is both expected and supported by other independent studies, old and recent [91, 14].

What is particular in these results, however, is that the classification does not degrade equally with respect to all performance measures. Rather than a random concept drift (like the users of a shopping site changing their preferences and thus affecting the results of a recommendation algorithm), we have here a directed change that fits a spam evasion pattern. It has been remarked that when Machine Learning is applied in adversarial settings such as spam detection, concept drift is biased towards evading detection [15], a process that can asymmetrically impact classification recall more than precision. The results we obtained in Chapter 3 support this claim. They show that the evaluated features are still capable of reliably capturing a part of the spammers population (with high precision), while they seem inefficient in distinguishing other spammers from the rest of the legitimate Twitter population.

We have suggested in the previous chapter that these qualities allow to exploit these systems for discovery of spammers in the wild. We also suggested that discovery can be used for seeding, a concept that has been successfully exploited in unsupervised systems (e.g. in Leas, YouTube's unsupervised detection system where seeds are used to start localized graph clustering [18]).

In this chapter, we build on the conclusions of the last chapter and expand the discovery and seeding direction to a full-fledged implementation of a social spammers detection on Twitter. The system we propose exploits two main assumptions:

- Discovery and seeding: when the target population (spammers) shifts, supervised classifiers can be used for discovery instead of detection. Discovery means to reliably uncover accounts that can be exploited as seeds in further detection systems.

- Users similarity: Accounts similarity implies class homophily. When two accounts are judged similar by an adapted definition of similarity, they are more likely to have the same class (i.e., be both spammers or both legitimate).

The first assumption requires a notion of prior belief that can be modeled with the prediction probability of a supervised classifier. The second assumption requires a notion of similarity graph that would allow belief propagation.

This combined belief/similarity framework is a good candidate to a probabilistic graphical model. We use here the Markov Random Field (MRF), an undirected graphical model that models joint probability over dependent random variables. The accounts are modeled as random variables, similarity is modeled as edges. Node potentials model prior belief and edge potentials model homophily[1] between connected accounts.

[1]Homophily is a term originally coined by the social scientist Paul Lazarsfeld to refer to a tendency of similar individuals to connect together. In Complex Networks, homophily refers to the theory that nodes with similar attributes are more likely to be connected than those that have dissimilar attributes.

The classification problem can then be cast as an inference over dependent variables, and learning would correspond to finding the MRF parameters that minimise the classification loss. In this chapter, we use the conventional loopy belief propagation as the inference algorithm and identify the MRF parameters with grid search. We leave the discussion of advanced inference and learning as well as alternative graphical models and loss measures to the next chapter.

In the same spirit of the evaluation undertaken in the previous chapter, we pay special attention to the particularities of implementation on Twitter, and evaluate the proposed system on the dataset previously described. It is important to note, however, that the proposed concept is platform-agnostic and can be re-formulated and adapted to any online social network.

Chapter structure. The remaining of this chapter is structured as follows. An informal introduction to probabilistic reasoning and probabilistic graphical models is provided in Section 4.2.1. Section 4.2.2 introduces an enriched formulation of the classification problem, its input and expected output. This formulation takes into account predictions of other state-of-the-art systems and exploits similarity between social accounts. Section 4.3 introduces the proposed system and explains its main components. We discuss in Section 4.4 how similarity between accounts is defined and the practical construction of similarity graphs. We also introduce the Markov Random Fields formulation and how belief propagation can be applied in the context of the proposed solution. Section 4.5 presents the experimental evaluation and compares the results with the baseline obtained in the previous chapter. We discuss the results and their implications in the conclusion.

Contributions. The main contributions of this chapter are summarized as follows:

- Undirected graphical models can be used to model the problem of social spam detection.
- The Markov Random Fields formalism allows a hybrid social spam detection model that exploits both users features and their similarity.
- A robust measure of similarity between users can be defined in terms of common content published by these users.
- The results validate that biased and inaccurate prior predictions on users classes can be effectively used in the context of probabilistic graphical models, as demonstrated by the significant increase in recall obtained by the proposed approach.

4.2 Probabilistic Modeling of Spam Classification

4.2.1 An Informal Introduction to Probabilistic Graphical Models

In Chapter 3, we discussed training classifiers to classify an individual account. This is a common setting in machine learning problems, where we often want to classify a

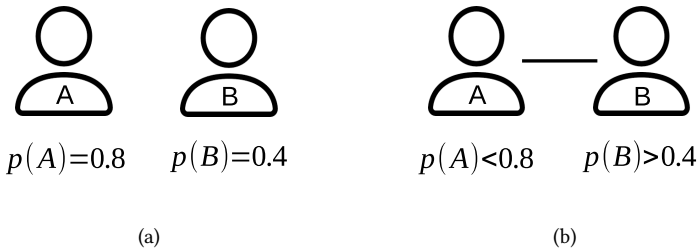

Figure 4.1: A toy example showing how having a connection between two accounts alters beliefs about their class probabilities.

single instance of data, e.g. a single image. An example of individual account classification is illustrated in Figure 4.1a. The classifier assigns a spammer class to account A and a legitimate class to account B. The prediction probabilities $p(A = spammer)$ and $p(B = spammer)$ are referred to simply in the figure as $P(A)$ and $P(B)$ respectively. They quantify the classifier's uncertainty. Now let us tackle the case where A and B are connected (Figure4.1b). Suppose for example that they exhibit similar behavior, share the same content or have common friends. Having this connection pointed out, account A looks less of a spammer since A is connected to account B, an account we believe is legitimate. On the other hand, our belief of B being legitimate is undermined by its connection to account A, an account we believe is a spammer. This example illustrates probabilistic reasoning. We intuitively think of these accounts as dependent, and knowing something about the class of one account should impact our belief on the class of the other. The simplest way to represent this "conditional" dependency is the Bayesian theorem stating the probability of B given A: $P(B \mid A) = P(B \cap A)/P(A)$.

The supervised framework discussed above treats each account independently and cannot be leveraged to update our belief of the classes of these accounts. The most we can take from such a classifier is the probability of each individual prediction. Instead, the framework needed to model the above problem is one where the inference is performed jointly over connected accounts. This requirement can be best captured by the framework of Probabilistic Graphical Models (PGMs). PGMs are statistical models that allow the representation of a joint probability distribution over a graph of dependent random variables.

Probabilistic Graphical Models encompass a family of models. These models are most often classified depending on whether the edges in the graphical model are directed or undirected. Directed Graphical Models (DGMs), usually referred to as Bayesian Networks (BN), are graphical models where edges are directed, indicating a cause and effect relationship between connected random variables (e.g. the difficulty of the course impacts the grade but not the opposite). Undirected Graphical Models (UGMs) or Markov Random Fields (MRFs) are the graphical branch that deals with undirected edges, encoding a symmetrical relationship between connected variables.

Since we usually think of relationships between accounts (especially the similarity relationship) as symmetrical, it follows that Markov Random Fields are a better fit to the modeled problem.

4.2.2 Formal Definition of the Spam Classification Task

Setting and Input. Assume we have a set of social accounts U, where each user $u \in U$ has an associated vector of numerical characteristics $\mathbf{x}_u \in X$ and a set of messages $M(u)$ that represents the content posted by this user. For each user u, we would like to assign a class y_u in $L = \{0, 1\}$ where 0 denotes a legitimate account and 1 denotes a malicious account. We also have access to an inaccurate oracle that predicts, for each account u, the probability $p(y_u = 1)$ that u is malicious based on $\mathbf{x}_u$.

Domain knowledge indicates that if users have content in common, they are likely to share the same class, i.e., be both legitimate or both malicious.

Goal. We would like to know if, given the biased individual predictions offered by the oracle, it is possible to reach a better individual prediction for each user by taking into consideration its similarities to other users. The similarity is formulated here in terms of content and the leading assumption is that the similarity can be used to calibrate the bias in the oracle's predictions.

Formal Definition. We formulate the problem of assigning a class to each user u as a classic classification problem where the goal is to find a mapping from the user's representation $\mathbf{x}_u$ to the set of labels L.

To take into account the similarity between accounts, we rely on the Markov Random Field formalism, which allows us to define dependencies between predictions of similar users. The model represents the class of each user u as a random variable Z_u and the relations between these variables as a graph $G(V, E)$, where $V = \{Z_u\}_{u \in \text{Users}}$ is the set of users predictions and E is the set of edges linking similar users.

The dependency between users is simplified by assuming the Markov property, defined by a node being independent of all other nodes given its neighbors. On the defined undirected graph, the local Markov property is formally stated as: $P(Z_u \mid Z_{V \setminus u}) = P(Z_u \mid Z_{\mathcal{N}(u)})$, where $\mathcal{N}(u)$ is the neighborhood of u.

4.3 Proposed MRF Solution

The problem we formulated above and the general system we propose to improve the classification performance are both platform-agnostic. The details of the solution's implementation below are specific to Twitter but the solution can be adapted to any social network platform in which a similar problem can be defined (e.g. Facebook, Instagram, etc...).

We propose here to represent the different components of the problem with the Markov Random Field formalism. Section 4.3.1 introduces a high-level overview of the components of the proposed solution. Section 4.3.2 discusses the details of the MRF components, namely, the mapping between the elements of the classification

task and the MRF model and inference based on Loopy Belief Propagation. The similarity graph construction is another important component of the proposed system. It is a subject that deserves a separate discussion that we leave to Section 4.4.

4.3.1 General Solution Design

The formal definition of the problem includes three main elements: Predictions of features-based classifiers and a graph of similarity as input, and classes of accounts as output. We incorporate these elements into the system as follows:

- Accounts classes are MRF nodes representing random variables.
- Similarity is used to create edges between accounts, thus introducing variables dependency, a concept that is lacking from the traditional supervised classification models.
- Predictions of the supervised classifiers are turned into prior beliefs over the model variables and updated through belief propagation. This ensures that the definition of each account as a vector of statistical features is still exploited by the current system's solution.

The data flow of the proposed system, illustrated in Figure 4.2, can be summarized as follows:

1. **Data Crawling:** Accounts and content information is first crawled from the online social platform.
2. **Features Extraction:** Distinguishing behavioral, social and content-based characteristics are extracted from accounts content and profiles, and a numerical features vector is assigned to each account.
3. **Priors Computing:** A *prior* probability is assigned to each account given its numerical features vector (in Section 4.5, we obtain priors via state-of-the-art supervised classifiers but other sources can also be used to assign a prior belief to the class prediction).
4. **Graph Construction:** To construct the users similarity graph, a bipartite graph of users and messages is created to identify accounts that have identical or very similar content (Section 4.4).
5. **Posteriors computing**: Joint optimization of labels is finally applied using Loopy Belief Propagation over the constructed Markov Random Field (defined in Section 4.3.2). Once the propagation converges, the most probable configuration of labels is inferred from the resulting posterior probabilities (Section 4.5.2).

We focus in the following on formally defining the Markov Random Field model, how to incorporate prior predictions and how to exploit similarity for belief propagation.

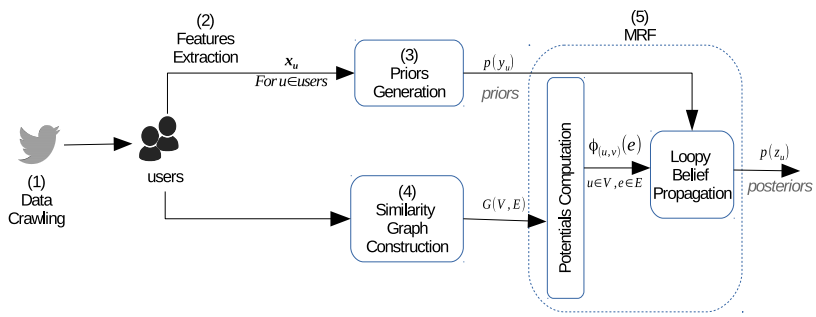

Figure 4.2: General architecture of the proposed system.

4.3.2 Markov Random Field for Social Spammers Detection

An MRF (G, Ψ) is a probabilistic graphical model that allows joint inference over dependent random variables. It consists of a graph $G(V, E)$, where:

- $V = \{Z_u\}_{u \in \text{Users}}$ is the set of random variables corresponding to users.
- $E = \{(u, v)\}$ is the set of edges denoting a dependency between two random variables Z_u and Z_v.

A set Ψ of potential functions govern the relationships between random variables. Potentials are factors defined over cliques of nodes. We use here the pairwise MRF model, which allows defining two types of potentials: edge (or pairwise) potentials and node (or unary) potentials.

4.3.2.1 MRF Potentials

Edge potentials are defined over edges in E. They ensure that the model responds to the smoothness criteria between connected variables in V, and generally direct the model towards predicting the same class for connected nodes. Unary potentials, on the other hand, are defined over individual nodes. They make it possible to take into consideration the features vector of each account by penalizing discrepancy between an observation vector $\mathbf{x}_u$ and the predicted class Z_u of user u. see Figure 4.3 for a graphical representation of potentials on a graph of users. We construct these potentials as follows:

(i) A unary potential ϕ_u is a local function that quantifies how favorable a class is for node Z_u given its features vector $\mathbf{x}_u$. Although an MRF potential is not a probability and can take any value in the set of real numbers, we define the unary potential here as a function that, for each user $u \in U$ and class in L, associates a probability:

$$\phi_u : U \times L \rightarrow [0, 1]. \tag{4.1}$$

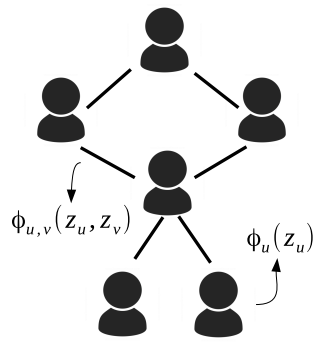

Figure 4.3: A graph of users illustrating node potentials defined over users and edge potentials defined over edges.

The unary potential is thus defined as a vector of two probabilities, the value of which is the system's prior belief about the class. This probability can be obtained from multiple sources including a supervised classification model trained on users features. This allows to indirectly incorporate the features information into the proposed MRF model as follows:

$$\phi_u(Z_u) = \begin{cases} 1 - p_u & \text{if } Z_u = 0 \\ p_u & \text{if } Z_u = 1 \end{cases} \tag{4.2}$$

where $p_u = p(y_u = 1 \mid \mathbf{x}_u)$.

(ii) An edge connects two nodes, Z_u and Z_v, if the corresponding users u and v are connected in the constructed similarity graph. Each edge is associated with a pairwise potential $\phi_{u,v}(Z_u, Z_v)$. In the current context, the edge potential is a function that represents compatibility between labels. Formally, edge potentials are defined as functions that for every realization of a pair of labels (in L), associates a real-valued factor quantifying its likelihood. Note that, in this implementation, the edge potential is the same for all edges and is not conditional on the observations. For the pairwise potential

$$\phi_{u,v} : L \times L \to \mathbb{R}^+, \tag{4.3}$$

We define the edge potentials as follows:

$$\phi_{u,v}(Z_u, Z_v) = \exp(f(Z_u, Z_v)) \tag{4.4}$$

Edge potentials are defined using a matrix of 4 parameters associated with the connection strength between classes of users, as defined in the following edge potential matrix:

$$\phi_{u,v}(Z_u, Z_v) = \exp\begin{array}{cc} & \begin{array}{cc} Z_v = 0 & Z_v = 1 \end{array} \\ & \begin{pmatrix} w_0 & w_2 \\ w_1 & w_3 \end{pmatrix} \begin{array}{c} Z_u = 0 \\ Z_u = 1 \end{array} \end{array} \tag{4.5}$$

$$\begin{pmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{pmatrix} \qquad \begin{pmatrix} e^{w} & 1 \\ 1 & e^{\alpha.w} \end{pmatrix}$$

(a) (b)

Figure 4.4: The edge potential matrix in (a) a symmetric special case, (b) an asymmetric case where inter-spammers and inter-legitimate connections are assigned different strengths.

where $w_0, w_1, w_2, w_3 \in \mathbb{R}$ and $Z_{u\ (resp.\ v)} = 1$ if u (resp. v) is a spammer.

We set $w_1 = w_2$ since they both designate a connection between a spammer and a legitimate user. The connection between two legitimate users and two spammers are governed by w_0 and w_3 respectively. There are two distinct cases that are generally used to model edge potentials:

- In a symmetric MRF, the edge potential is the same for edges connecting spammers (w_3) and edges connecting legitimate users (w_0). A common setting is to set $e^{w_1} = e^{w_2} = \epsilon = 0.1$ and $e^{w_0} = e^{w_3} = 1 - \epsilon = 0.9$ (see the associated matrix in Figure 4.4a).
- An asymmetric MRF provides a flexible relation between parameters (see Figure 4.4b). We set $w_0 = w$ and $w_3 = \alpha.w$, where α is a positive tunable parameter. As we demonstrate in Section 4.5, this gives our model a greater expressiveness and allows it to more accurately capture the empirical relationships in the dataset. Since the model is over-parametrized, we set $e^{w_1} = 1$.

4.3.2.2 Computing Marginal Probabilities by Loopy Belief Propagation

Our goal is to obtain posterior class probabilities over nodes, given the node and edge potentials of the defined MRF. The joint probability is defined as

$$P(Z) = \frac{\widetilde{P}(Z)}{\sum_{Z' \in L^N} \widetilde{P}(Z')}, \tag{4.6}$$

where

$$\widetilde{P}(Z) = \prod_{u \in V} \phi_u(Z_u) \prod_{(u,v) \in E} \phi_{u,v}(Z_u, Z_v).$$

An exact inference, that is computation of the marginal probabilities over the random variables, requires summing the joint probability (4.6) over all possible labels permutations and is intractable for large graphs. Additionally, since the graph contains loops, efficient inference algorithms designed for trees and chains are not applicable.

The Loopy Belief Propagation (LBP) algorithm [119] is an iterative message-passing algorithm that is frequently used to solve the inference problem on MRFs with general graph structure (e.g. in Computer Vision applications [120]). For graphs containing loops, LBP provides an approximate solution to the inference problem. LBP is considered linear in the number of edges. Its computational complexity is

$\mathcal{O}(d\,|E|)$, where d is the number of iterations required until convergence and $|E|$ is the number of edges. The outline of the algorithm is provided in Algorithm 1. Although the algorithm does not offer convergence guarantees, it converges in practice after few iterations [119]. Convergence is reached when beliefs converge, namely the inter-iteration difference is below a set threshold.

Algorithm 1: Loopy Belief Propagation algorithm

Input: MRF (G, Ψ), $\mathcal{E}$
$\Psi : \phi_u(Z_u), \phi_{u,v}(Z_u, Z_v)$.
$\mathcal{E} = \{(u, v), (v, u)\}$ for $(u, v) \in E$.
Output: Posterior marginal probabilities, $p(Z_u)$ for $u \in V$

```
1  /* Initialization                                                    */
2  m_{u→v}(Z_v) = 1 for (u, v) ∈ E /* Messages uniformly initialized
     */
3  b_u(Z_u) = 1 /* Beliefs of all nodes initialized to 1               */
4  repeat
5      /* Update messages                                               */
6      for (u, v) ∈ E do
7          m_{u→v}(Z_v) = Σ_{Z_u} ( φ_u(Z_u) φ_{u,v}(Z_u, Z_v) Π_{i∈Ne(u)\v} m_{i→u}(Z_u) )
       end
8      /* Compute node beliefs                                          */
9      for u ∈ V do
10         b_u(Z_u) ∝ φ_u(Z_u) Π_{v∈Ne(u)\v} m_{v→u}(Z_u)
       end
   until convergence;
```

4.4 Constructing a Users-similarity Graph

To better understand how similarity impacts accounts classification and how it can be integrated into the design of the solution, let us imagine the following scenario: two accounts A and B are known to be similar (the exact definition of similarity is not relevant in this particular context). A is flagged by a certain mechanism (also irrelevant for the purpose of this example) as a spammer. Let us also consider that $p(A)$ (resp. $p(B)$) represents the probability of A (resp. B) being a spammer, and that we have a prior distribution of the probability (for example a uniform distribution of $\alpha\%$, corresponding to the percentage of spamming accounts on the OSN). Knowing A's label, it is normal to assume that the posterior probability that B is a spammer would increase. This is because the similarity between A and B creates a dependency between the two variables that makes $p(B/A)$ different from $p(B)$.

4.4.1 Representing Similarity on OSNs

4.4.1.1 Similarity on Social Graphs

On online social networks, the concept of similarity between accounts has been traditionally captured by the who-connects-to-whom network in bidirectional networks (e.g. the friendship network on Facebook) or the who-follows-whom networks in unidirectional networks (e.g. on Twitter) [108, 107]. These networks, commonly known as the "social graphs" can be used as similarity graphs by assuming "strong trust" between connected accounts. The "strong trust" assumption is the assumption that friendship between two accounts means that there is a bidirectional endorsement between these two accounts. This assumption was mainly used in the literature in conjunction with the assumption that Sybils and legitimate accounts form distinct dense clusters. Edges are thus concentrated inside these clusters while few "attack edges" branch from one cluster to the other (see Figure 4.5).

Recent empirical analysis, however, suggests that the strong trust assumption is violated on unidirectional social networks. This is especially the case for Twitter [48]. On Twitter, following cannot be considered a two-sided endorsement, and thus if a spammer follows a legitimate account, his account should not become more trustworthy as the "strong trust" assumption implies. Additionally, even when the relationship is reciprocated (A follows B and B follows A), this can be due to the high reciprocity of the human interaction behavior as we have mentioned in Chapter 2. This is also complicated by accounts of celebrities and companies as well as social butterflies (also previously discussed in Chapter 2) sometimes adopting this behavior as their default behavior (i.e., following back all their followers). All of these observations explain how it is feasible for spammers to integrate the legitimate graph and why the notion of "strong trust" is seriously flawed on OSN (as opposed to e.g. peer-to-peer networks where connection between accounts encodes higher levels of endorsement [96]). A similar situation has also been reported on RenRen, the Chinese OSN [101].

4.4.1.2 Similarity on Interaction Graphs

Recent unsupervised detection systems on Facebook (CopyCatch [16] and Synchro-Trap [17]) and YouTube (LEAS [18]) have used interaction graphs instead of social graphs (see an illustration of these interaction graphs in Figure 4.6). We propose a similar idea for Twitter where we base similarity on a bipartite content-users graph. The assumption here is that complicit spammers need to share the same content for better coverage of their targeted audience. Shared content is also a more significant *complicity* signal than an unsolicited following link on Twitter.

In choosing this definition of similarity, we rely on the assumption that accounts that have common content tend to belong to the same class of users (see Figure 3.3 for a tweet that was shared across many profiles on Twitter). Specifically, spammers belonging to the same or similar spam campaigns tend to have similar content. We start by defining a bipartite users-to-messages graph and then collapse this bipartite graph into a users-similarity graph. We describe some special considerations to take into account when applying this general mechanism to Twitter. We show that the

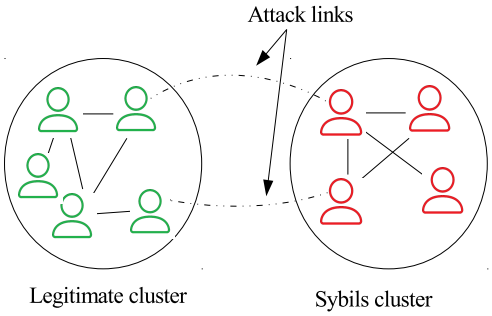

Figure 4.5: Works based on the strong trust assumption, model the Sybils detection problem as two dense clusters of legitimate and Sybil accounts and assume that edges between these two clusters are attack edges.

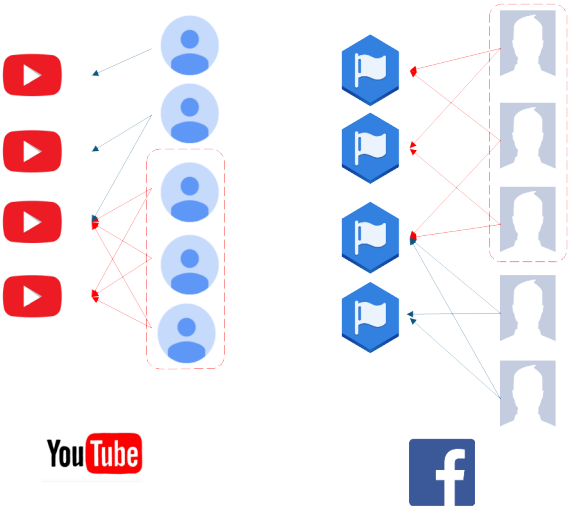

Figure 4.6: Bipartite interaction graphs on YouTube (accounts commenting on videos) and Facebook (accounts liking pages). Dense bipartite clusters may indicate malicious activity.

proposed similarity measure successfully captures the concept of class homophily between social accounts.

4.4.2 Proposed Content-based Similarity

We explain in this section the graph construction mechanism that we propose to use as a way to capture similarity between social accounts. The previous sections have demonstrated the difference between the two major approaches of building accounts

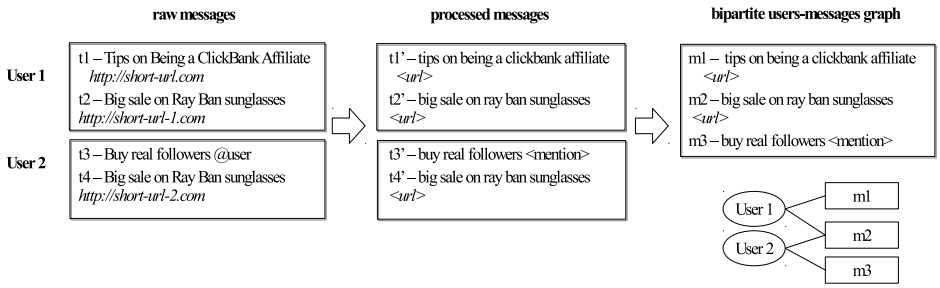

Figure 4.7: Construction of the bipartite users-messages graph on a toy example.

graphs on online social networks. We have discussed the merits of the interaction-based graphs and we propose here to build interaction on a bipartite users- messages graph. The goal is to measure collusive behavior using posted content. This is a strong signal that cannot be generally interpreted as transient similarity but is often due to accounts acting together.

4.4.2.1 Bipartite Users-Messages Graph

To construct the bipartite users-to-messages graph, we start by processing the text messages published by each account and create an edge between accounts and the processed text of their messages. The process is illustrated in Figure 4.7.

Text pre-processing. Adding spurious characters, obfuscating final urls and varying the user mentioned in messages are all techniques that abusers are known to use to avoid their messages being detected as exact duplicates. Text pre-processing is therefore a vital part of the users-messages graph construction pipeline. In this instance, it consists of lowercasing and tokenizing texts and removing punctuation. Urls and users mentions are replaced by place holders (i.e., <url> and <mention> respectively). We do not replace hashtags by place holders since they can often be used as an integral part of the message text (due to the limitation on the number of characters per tweet). The described processing is done once for each message by each user in the dataset, and the complexity of this step is linear in the number of messages.

Short texts containing less than three non place holders tokens (e.g. "Hi <mention>") are discarded to avoid creating false connections between users. The remaining exact replicates are merged and the set of resulting unique messages $\mathcal{M}$ forms the messages in the bipartite users-messages graph.

4.4.2.2 Users Similarity Graph

The users graph is generated by collapsing the messages in the bipartite graph. This is done by creating an edge between every pair of users that are connected to the same message as detailed in Algorithm 2. Table 4.1 explains the notation used in

Table 4.1: Notations used for bipartite and similarity graphs.

Symbol	Description
U	set of users
$\mathcal{M}$	set of processed messages
u, v	users in U
m	a message in $\mathcal{M}$
n_m	number of users that posted message m
E_{UM}	set of bipartite user-message edges
(u, m)	a user-message edge in the bipartite graph
E_U	set of user-user edges in the users similarity graph
$G(U, E_U)$	users similarity graph
$G(U \cup \mathcal{M}, E_{UM})$	bipartite users-messages graph

the algorithm. This is a time-consuming process since the complexity of generating all users pairs is quadratic in the number of edges between each message and its associated users. The number of users pairs is specifically equal to:

$$\frac{1}{2} \sum_{m \in \mathcal{M}} n_m(n_m - 1), \tag{4.7}$$

where $\mathcal{M}$ is the set of processed messages and $n_m =| \{(u, m) \in E_{UM} \text{ for } u \in U\} |$ is the number of edges in the bipartite graph that link to message m.

Note that despite its high computational load, the generation of a users-similarity graph is a prerequisite of many unsupervised detection models [18, 17]. It can be partially parallelized by assigning pairs generation of each message to a different process. A discussion of the parallelized implementation of a similarity graph on Facebook (based on login information and IP addresses) is provided in [17]. A threshold can be implemented in order to prevent the generation of pairs for highly popular content (messages in our case) [18, 17]. These messages are usually associated with legitimate content and the number of users linked to them would yield a significant computational load. Since the number of users associated with messages in our dataset is reasonable (most popular message is shared by 24 users, see Table 4.3), we did not have to implement a similar measure.

4.4.2.3 Twitter-Specific Considerations

The main assumption regarding the resulting bipartite graph described above is that created edges encode homophily: a malicious account creates spam messages, while a legitimate account creates legitimate messages. While applying the general graph construction mechanism described above to Twitter, however, we became aware of two special cases in which an edge between a user and a message can be used to falsify credibility.

1. Content copying: A malicious account can engage in legitimate content copying. This leads to legitimate content (endorsed by links from legitimate ac-

Algorithm 2: Users pairs generation from the bipartite users-messages graph

Input: The set of processed messages $\mathcal{M}$ and the set of bipartite edges E_{UM}
Output: $E_U = (u, v, w)$ the set of weighted edges in the users similarity graph where $u, v \in U$ and w is an integer weight.

```
1  D ← new hash map
2  for m ∈ M do
3      U_m ← array of {u ∈ U | (u, m) ∈ E_UM}
4      for i = 1 to |U_m|−1 do
5          for j = i + 1 to |U_m| do
6              u ← U_m[i], v ← U_m[j]
7              s ← min(u, v)
8              d ← max(u, v)
9              if (s, d) ∈ D then
10                 D[(s, d)] ← D[(s, d)] + 1
11             else
12                 D[(s, d)] ← 1
           end
       end
   end
13 E_U ← {u, v, D[(u, v)]} for (u, v) ∈ D
```

counts) being linked to malicious accounts, thus boosting the credibility of these accounts.

2. Compromising legitimate accounts: This has the opposite effect of content copying. When a spammer gains control of a legitimate account, any malicious or spam content the spammer publishes using this account will be endorsed by the previously established legitimacy of the compromised account.

Both problems can be solved using the notion of application on Twitter. An application, also known as the tweet source, is a term used to coin the software that published the tweet. Each application has a unique text identifier. We introduce the following changes on the algorithm described above:

1. Content copying: we identify a unique message by both its processed text and the source that published it. Thus, even if two messages share the same text, they are considered as different entities if they were not published by the same source. This restriction is reasonable since most legitimate accounts use the web interface or Twitter's mobile applications (e.g. Twitter for Android and Twitter for iPhone), while automated accounts use content management applications (e.g. dlvr.it and buffer) or custom scripts. Moreover, malicious campaigns often work in bursts to accomplish maximum visibility, and thus the shared content is usually published by the same application.

2. Exploiting compromised accounts: we introduce the notion of applications profiles. These are computed by extracting the application used to post each

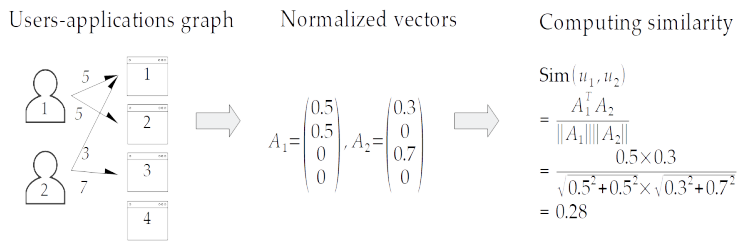

Figure 4.8: A toy example showing the computation of similarity based on applications profile.

tweet and computing, for each account, the normalized proportion of tweets posted by each application. Since temporarily compromised accounts are often quickly restored, we expect that a compromised account that has a malicious message in common with a malicious account, will nonetheless have a significantly different application profile compared to this latter. To quantify this difference, we compute the cosine similarity between application profiles. For two users u and v, this is defined as the normalized inner product of the normalized applications vectors A_u and A_v (as illustrated in Figure 4.8). The similarity is defined as:

$$Sim(u, v) = \cos(A_u, A_v) = \frac{A_u^T A_v}{\|A_u\| \, \|A_v\|}, \tag{4.8}$$

where $\|\cdot\|$ is the Euclidean norm.

4.4.2.4 Implementation

We implemented the revised version of the graph construction algorithm by defining messages as a tuple that contains both the processed text and the source application (Table 4.2 lists the top 10 applications generating 87% of the messages in the ground-truth dataset). We then filtered edges in the resulting users graph according to the pairwise similarity of applications profiles. We removed edges that have a content weight of one (one common tweet) or an application similarity rate of less than 0.9. This choice is taken so that an edge represents real complicity between users. It also decreases the probability of linking two users based on text that is falsely identified as similar.

The resulting similarity graph is a sparse graph with 157 nodes and 549 edges. The connected nodes represent 20% of the number of accounts in the ground-truth dataset. Edges represent 4.5% of the number of edges in a fully connected graph with 157 nodes.

The similarity graph in Figure 4.9 illustrates the labels homophily captured by the similarity measure. It shows that linked users generally belong to the same class. Legitimate users and spammers also tend to form their own respective clusters. Among

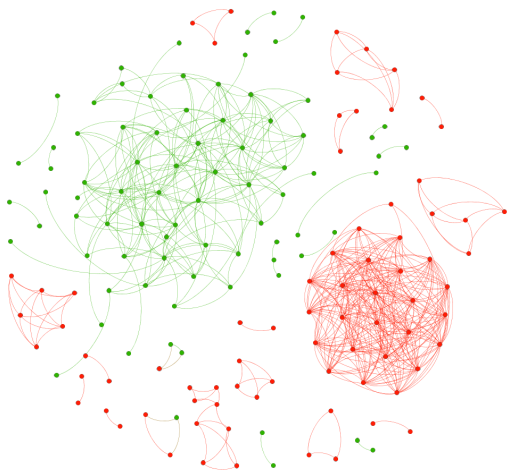

Figure 4.9: The similarity graph of connected users in the ground-truth dataset. Legitimate users are represented in green while spammers are shown in red.

Table 4.2: Most used applications in the groundtruth dataset (in terms of the number of unique messages).

Application	**No. messages**	**%**	**Application type**
Twitter for iPhone	71,755	39.1	Twitter affiliated
Twitter Web Client	27,397	14.9	Twitter affiliated
Twitter for Android	17,866	9.7	Twitter affiliated
TweetDeck	11,198	6.1	Content management (Twitter affiliated)
Done For You Traffic	7,963	4.3	Content management
dlvr.it	7,079	3.9	Content management
IFTTT	6,343	3.5	Content management
Hootsuite	4,994	2.7	Content management
Google	3,572	1.9	Content Referral
Facebook	2,204	1.2	Content Referral

the 30 identified clusters, only 2 contain spammers and legitimate nodes simultaneously. This validates that the proposed similarity measure does indeed result in users of the same class being linked together. The high modularity (0.873) and average clustering coefficient (0.795) of the graph also demonstrate that users tend to cluster in communities of mutually similar users that are quite distinct and disassociated from the rest of the graph. Moreover, the graph clearly shows that the assumption that spammers form one connected community, which forms the basis of many previous works, does not hold.

Table 4.3: Processed texts and applications of the top five tweets in the dataset (in terms of number of users sharing the tweet).

Processed text message	Application	No. users
“<url>the daunting risks of laparoscopic obesity surgery”	Done For Your Traffic	24
(translation) "<mention>: to al nasr fans [sports team], I was honored this evening to be one of the world championship players..."	Twitter for iPhone	7
“<url>15 reasons to join affiliate programs”	Done For Your Traffic	7
“<url>8 ways to improve your affiliate marketing strategies”	Done For Your Traffic	7
“<url>finding the perfect product at clickbank”	Done For Your Traffic	7

4.4.2.5 Computational Complexity of Graph Construction

We have discussed in Section 4.4 the computational complexity of generating users edges from a list of users associated with a post. In practice, there are two facets associated with the computational load of generating the graph:

- Quantifying the computational load of creating a similarity graph.
- Fixing a threshold to the number of users associated with a post (a unit of content).

Computational load of generating a similarity graph. We start by answering the first part: assessing the time required to build a similarity graph. For that, we need two information:

- An empirical distribution of the number of users per post (text/unit of content).
- An empirical estimation of the time required to expand a list of users as a function of the number of users in the list.

Recall that expanding a list of users refers to creating an edge between all pairs of users. The computational complexity of this procedure is $\mathcal{O}(n^2)$, where n is the number of users in the list. To assess the expansion time, we conducted a simulation on an Intel i7 machine with a 2.5 GHz clock and 16 GB of RAM and measured the time required to list all edges associated with a given number of users. We used a user ID format comparable to that of Twitter to mimic realistic memory usage. The results are shown in Figure 4.10 and confirm that the time increases quadratically in the number of users. Figure 4.11 shows the evolution of the logarithmic time as a function of the number of edges. It further confirms that the time is proportional to the squared number of users. The processing time reaches one minute around 10000 users (or 10^8 edges). This means that it becomes 10 minutes for $31k$ users and 100 minutes for $100k$ users. Note that for expanding less than 1000 users, the processing time of less than 1 second is negligible.

To assess the time required to build the similarity graph, we also need to assess the empirical distribution of the number of users to expand per post. For that, we

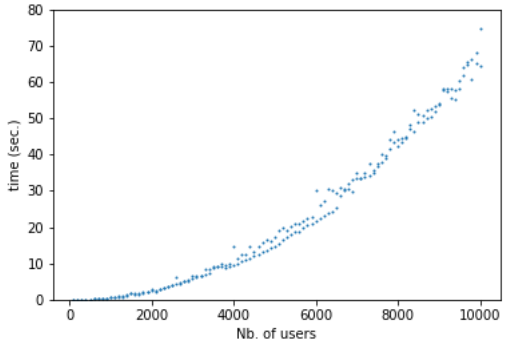

Figure 4.10: Time required to expand a list of users into edges.

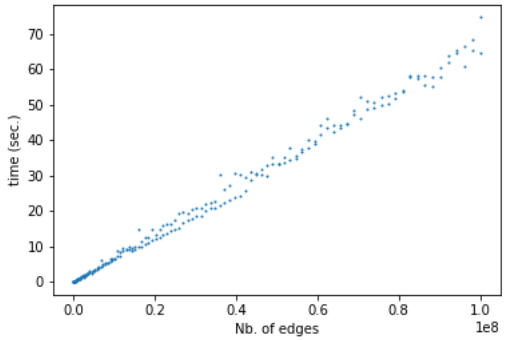

Figure 4.11: Time required to generate users edges.

consider the real case scenario where our system is used to detect spammers in trending topics. Trending topics are typically related to current and controversial topics and events and are characterized by a large audience. This makes them a particularly interesting target for spammers and opinion manipulators. We consider a dataset of trending topics we previously collected. The dataset covers trending topics in Kingdom of Saudi Arabia in the period between 19/3/2015 and 1/4/2015, and contains 1,124,926 unique tweets.

We construct the bipartite users-content graph and plot the distribution of users per text (processed post) for each trending topic as shown for example in Figure 4.12. All the distributions for the studied trending topics represent a similar pattern where every post (unit of content) is predominantly shared by a small number of users. This means that the associated users edges should be obtained in milliseconds. Only a few texts are shared by thousands of users. Specifically, a typical trending topic does not have more than 5 messages counting more than one thousand users. The most shared text in the studied dataset has been shared by 14k users and associated edges should

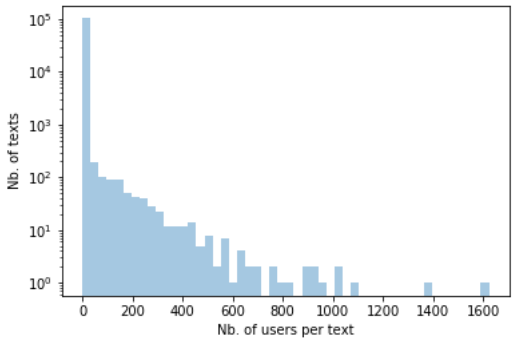

Figure 4.12: Histogram of the distribution of identical texts in a trending topic.

therefore be produced in around one minute.

Note that the construction of the graph is completely parallelizable in that every list of users (associated with a given post) can be assigned to a different machine core or mapper (in a MapReduce framework). The total time is therefore bounded by the time required to process the largest list of users. Even when the processing is serialized, the total time is dominated by the same value as the distribution is usually biased towards less popular posts.

Choosing a threshold. The question of where the threshold should be placed depends on the application and the available processing power. A threshold is the number of users beyond which a list of users is not expanded (not transformed into edges). We empirically assessed the upper bounds of content-based aggregation of users in trending topics and deduced that they can be processed in reasonable time. By interpolating the plot obtained in Figure 4.11, we can safely consider that a threshold of around $30k$ users is a reasonable restriction. Posts with this degree of popularity are usually initiated by celebrity profiles, and we can safely consider that a (text, application) tuple having $30k$ accounts is an indication of an organic legitimate sharing activity.

4.5 Experimental Evaluation and Discussion

We evaluate in this section the performance of the proposed MRF-based model over the ground-truth Twitter dataset. We compare these results to the baseline performance yielded by state-of-the-art supervised classifiers and discuss their significance and implications.

As we previously discussed in this chapter's introduction, one of the leading motivations of the system design is to establish the usability of supervised classifiers as tools for discovery and seeding. This allows to leverage the large supervised literature that characterizes spammers based on their behavioral, content and social network attributes. This is done by alleviating the main drawback of supervised

approaches – their degrading performance, by taking advantage of the high confidence of their spammers predictions. We show that, even with beliefs produced by a weak supervised classifier, the MRF model can leverage the prior predictions of the supervised classifiers and output improved predictions.

4.5.1 Experimental Setting

We use the same sets of features on which we evaluated the supervised classifiers in Chapter 3 to produce prior predictions and as a baseline to which the MRF-based predictions are compared. We use the 157 accounts belonging to the graph constructed in 4.4.2.4 to evaluate the MRF model with the prior probabilities predicted by the supervised classifiers. The remaining 610 accounts form the training dataset for the supervised classifiers (80% of the ground-truth dataset).

Generating Prior Predictions. We use the same sets of features reported in Table 3.8 in Chapter 3 (namely our selected set of state-of-the-art features and the sets of features proposed in Benevenuto and Stringhini), and train the same Machine Learning models (namely Support Vector Machines (SVM) [115], Logistic Regression (LR) [116] and Random Forests (RF) [117]). For each user u, with features vector denoted $\mathbf{x}_u$, the classification model predicts a class y_u with a probability[2] $p(y_u \mid \mathbf{x}_u)$. This probability quantifies the classifier confidence of its prediction and is the prior used to initialize beliefs before belief propagation. Figure 4.13 illustrates the beliefs computed by each of the supervised models for a cluster of abusive users.

MRF Implementation. We implemented MRF using the *UGM* library [121] in Matlab. For inference, we used the library's implementation of Loopy Belief Propagation (LBP). To find the best MRF parameters, we implemented grid search over these parameters.

Loopy Belief Propagation. Starting from the priors computed on each node, we applied LBP over the graph and updated classes beliefs according to the defined edge potentials. The MRF predictions are associated with posteriors obtained on LBP convergence.

Figure 4.14 illustrates how the predictions of a weak local classifier can be exploited to enhance the detection performance. It shows belief propagation on a cluster of 3 spammers. Edge potentials are computed for $\alpha = 2$ and $w = 0.6$ (see the next paragraph for more details on the choice of α and w). Because it is linked to two spammers, user 3 is correctly classified by MRF as a spammer. Moreover, users 1 and 2, being both linked and initially believed to be spammers, reinforce the prediction of each other, thus the probability of predicting the spammer class increases.

[2]In our implementation, we used the *predict_proba* function of the *scikit-learn* Python package to compute the probability each supervised model assigns to its prediction.

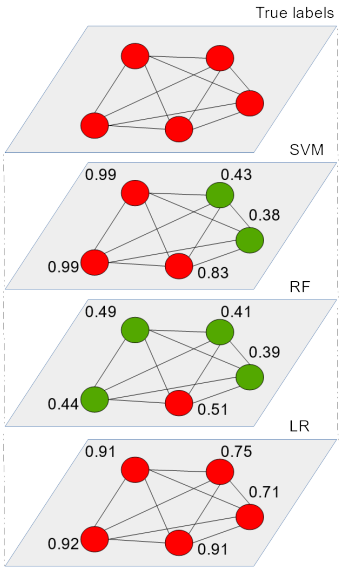

Figure 4.13: Prior beliefs computed by the supervised classifiers over a cluster of spammers. Nodes predicted as legitimate users are represented in green, while red represents nodes predicted as spammers.

4.5.2 MRF Classification Results

We discuss in this section the classification results obtained by optimizing the MRF parameters on the Twitter dataset and compare the results to the baselines established by the supervised classifiers in the previous chapter. Classification results of the MRF models are reported in Table 4.4. For each set of features, we compare the classification performance obtained by applying features-based classifiers, symmetrical MRF and asymmetrical MRF as defined in Section 4.3.

It is clear that the asymmetric formulation of MRF manages to outperform both the baseline performance and that of the symmetric formulation of MRF. And while this result is notable, it is also worth noting that the symmetric formulation of MRF, which is, to the best of our knowledge, the only formulation used in related applications (e.g. in applications on Amazon [106] and eBay [104]), does not in general improve upon the baseline performance. With the exception of the case where priors are obtained by training Logistic Regression on Stringhini features, the symmetric MRF formulation consistently yields an F1-measure that is inferior to the baseline. This justifies the need to optimize the graphical model's parameters empirically rather than based on a predefined symmetric assumption.

Performance of the Asymmetric MRF. The average gain in performance (recall and precision) achieved by the asymmetric MRF classification compared to the baseline local classifiers is shown in Figure 4.15 for α ranging between 2 and 3.5. In

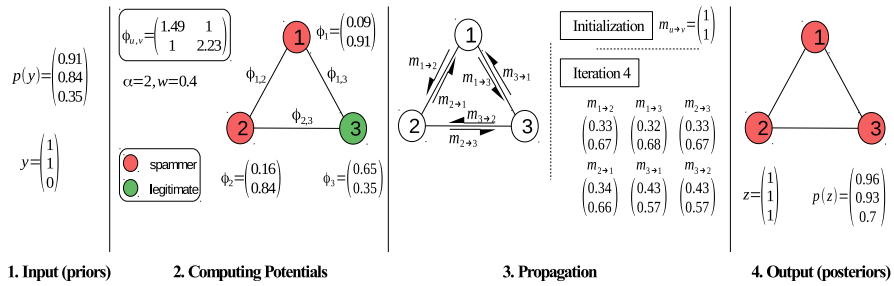

Figure 4.14: Loopy Belief Propagation illustrated on a cluster of 3 spammers. Input priors and potentials are shown on the left. The central frame shows the first and last iterations of belief propagation. The output frame shows posterior probabilities computed from the final values of messages. The algorithm in this instance converges after 4 iterations.

Table 4.4: Classification performance, evaluated over the test dataset, of the baseline supervised classifiers, the symmetric MRF classifier ($e^{w_0} = e^{w_3} = 0.9$ and $e^{w_1} = e^{w_2} = 0.1$) and the asymmetric MRF classifier ($w = 0.6, \alpha = 2.5$).

		All features			**Benevenuto features**			**Stringhini features**		
		Sup.	Sym. MRF	Asym. MRF	Sup.	Sym. MRF	Asym. MRF	Sup.	Sym. MRF	Asym. MRF
SVM	Precision	0.891	0.917	**0.919**	0.939	**0.955**	0.930	0.941	**1.000**	0.966
	Recall	0.598	0.571	**0.883**	0.756	0.273	**0.857**	0.195	0.143	**0.364**
	F1	0.715	0.704	**0.901**	0.838	0.424	**0.892**	0.323	0.250	**0.528**
	Accuracy	0.752	0.764	**0.904**	0.847	0.637	**0.898**	0.573	0.250	**0.682**
LR L1	Precision	0.865	0.860	**0.890**	0.961	**1.000**	0.924	**1.000**	**1.000**	**1.000**
	Recall	0.549	0.558	**0.844**	0.598	0.545	**0.792**	0.159	0.325	**0.506**
	F1	0.672	0.677	**0.867**	0.737	0.706	**0.853**	0.274	0.490	**0.672**
	Accuracy	0.720	0.739	**0.873**	0.777	0.777	**0.866**	0.561	0.669	**0.758**
LR L2	Precision	**0.956**	0.900	0.933	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Recall	0.524	0.468	**0.727**	0.317	0.182	**0.714**	0.159	0.325	**0.481**
	F1	0.677	0.615	**0.818**	0.481	0.308	**0.833**	0.274	0.490	**0.649**
	Accuracy	0.739	0.713	**0.841**	0.643	0.599	**0.860**	0.561	0.669	**0.745**
RF	Precision	**0.955**	0.924	0.902	**1.000**	0.933	0.928	0.960	**1.000**	0.925
	Recall	0.780	0.792	**0.961**	0.585	0.727	**0.831**	0.585	0.532	**0.805**
	F1	0.859	0.853	**0.931**	0.738	0.818	**0.877**	0.727	0.695	**0.861**
	Accuracy	0.866	0.866	**0.930**	0.783	0.841	**0.885**	0.771	0.771	**0.873**

this plot, gain is defined as the absolute change between the MRF performance and the performance of the local classifier on which it is based (prior predictions). The evolution of performance is computed as a function of edge potentials ($w_1 = w$ and $w_2 = \alpha \times w$). The figure clearly shows that MRF classification consistently increases the recall while maintaining precision around its baseline level.

Figure 4.16 shows the performance gain for $\alpha = 3.5$. Note that the best performance is obtained by setting w between 0.4 and 0.7. This yields a positive increase in recall (20 to 27% on average) while maintaining original precision (average decrease

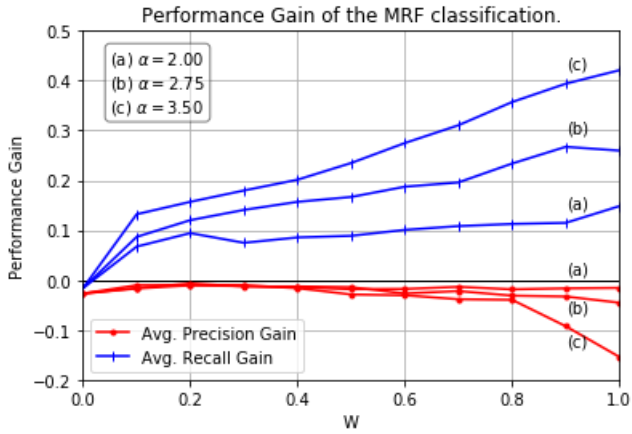

Figure 4.15: Performance gain of the MRF classification as a function of edge potentials for α ranging from 2 to 3.5.

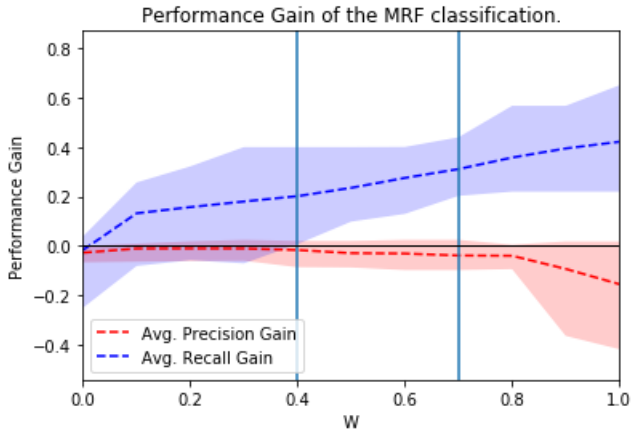

Figure 4.16: Average absolute gain in performance as a function of edge potentials for $\alpha = 3.5$. Upper and lower limits correspond to the maximum and minimum gain at each value of w.

of 1.6 to 3%).

Asymmetric MRF Parameters. The reported values of α and w are in agreement with the empirical characteristics of the dataset for several reasons:

- Since legitimate users are more likely to connect to legitimate users and spammers are more likely to connect to spammers, it is to be expected that w_0 (resp. w_3) should be higher than $w1$. The strength of connection between legitimate users (resp. spammers) should be bigger than that of a spammer to legitimate

user connection. In the dataset, users having the same class are more likely to be connected than users of different classes (only 4 edges in the similarity graph are inter-classes edges).

- The asymmetric formulation allows w_3 and w_0 to have different values. The grid search discussed above shows that performance is enhanced by setting w_3 higher than w_0. Recall that w_0 is assigned to a legitimate to legitimate connection and that w_3 is assigned to a spammer to spammer connection. This can be explained by spammers being more densely connected than legitimate users. Another reason is that classifiers are always more confident about their spam labels (see the precision of the spam class in Table 4.4) making a spam label prediction more trustworthy than a legitimate label prediction.

4.5.3 Discussion and Generalization Insights

We present in this section the generalization insights gained from the results of the model above and discuss practical implementation issues that are faced in a large-scale implementation.

4.5.3.1 Modularity vs. Homophily

For the MRF to work, edges should generally indicate a relationship of homophily: connected nodes have the same class. This assumption is vital to ensure that belief can be propagated on the users similarity graph. As discussed in Section 4.4, the graph construction mechanism is tuned so that the number of edges connecting accounts from opposite classes is minimized. Although the resulting graph is modular, modularity in itself is not seeked: if two spam (resp. legitimate) clusters become connected through an edge, the model will become even more certain about its posterior predictions. In this case, an edge offers additional information.

The opposite case is problematic. If a legitimate account or cluster of accounts become connected to a spam cluster, belief will be propagated between the two clusters, leading to a decreased certainty in both the spam and legitimate class predictions. It is therefore important for the graph construction mechanism to result in homophilic edges and for the similarity measure to connect accounts having the same class.

A practical obstacle that would be faced in the case of a large-scale implementation is represented by “quotes apps”. These are applications that generate automated sayings and posts and are generally subscribed for by both legitimate and spam accounts. The latter benefit from these applications in keeping their accounts active and posting. Tweets posted by these applications on behalf of subscribing accounts will inevitably result in edges created between spam and legitimate users. A simple solution would be to filter posts generated by these applications. This is feasible as these applications have a large throughput and are usually easy to identify when aggregating content from a large collection of users.

4.5.3.2 Effect of baseline recall and precision

All compared baseline classifiers have a relatively high precision and can therefore be reliably used to detect seeds of spam accounts. Table 4.4 shows that, even when baseline recall is lower than 50% (Stringhini features), beliefs can be effectively propagated, and the MRF model can increase the recall while maintaining precision. This can be explained by two reasons. The first is that the edge potentials matrix favors spammer-spammer edges. When a spammer is identified, connections are more likely to be predicted by the model as spammers. The second reason is that accounts features are not randomly distributed among users. Connected users are more likely to have similar features and therefore to have similar prior predictions. In other terms, when seeds are discovered, they are more likely to be clustered together than to be randomly distributed among spammers clusters. Thus, even when the baseline recall is low, the concentration of seeds in particular clusters ensures that other spammers in those clusters will be correctly identified.

4.5.3.3 Role of the edge potentials matrix

MRF is a generative model. The potentials can be used to quantify the likelihood of incidence of a particular edge configuration. Higher values of w would therefore indicate that a spammer-spammer edge is much more likely than other configurations. This blocks belief propagation across the graph as inference becomes dominated by the edge potential. Figure 4.15 shows a general trend of decreasing precision when w reaches higher values. This can be explained by the edge potential becoming significantly higher than the node potentials[3]. Thus, the model becomes mostly equivalent to an MRF with no observations on the nodes and assumes that most edges are statistically associated with spammers. This is similar to the case where a traditional classifier assumes that all or the majority of classified instances belong to a certain class, resulting in a perfect recall and a low precision.

Spam clusters are typically denser and are therefore associated with more edges. This makes asymmetric edge potentials matrices better at capturing the distribution of edges in the users graph. However, Lower values of w should be preferred to avoid the scenario discussed above.

4.6 Conclusion

A main point to underline in this chapter's results is that similarity, when used in a probabilistic framework, can improve the performance of a weak local classifier.

The similarity we proposed in this chapter is issued from a behavioral dynamic that is hard to change without spam losing its utility. Therefore, similarity represents a facet of the solution that is stable and robust to changing spammers characteristics. The results showed that, when classification precision is high, it is possible to improve recall for a wide range of baseline recall values. We have assumed that the trained features-based classifiers act as surrogates for classifiers for which performance has deteriorated due to spam evolution. The implication of this in the context

[3]For $w = 1$ and $\alpha = 3.5$: $\phi_u \in [0, 1]$, $\phi_{u,v}(1, 1) \approx 33$, $\phi_u \ll \phi_{u,v}(1, 1)$.

of the above results is that the performance of a features-based classifier can be improved (for relatively low levels of recall) or restored (for relatively high levels of recall) in the presence of spam evolution.

These results represent an important step towards the goal of leveraging accounts dependency for exploiting scarce, inaccurate and biased predictions from variable sources (including biased statistical classifiers) and towards building detection systems that are more robust to features variations.

The empirical evaluation of the proposed model on the ground-truth Twitter dataset showed that the symmetrical MRF formulation, which is routinely used in various applications, did not improve the classification performance compared to baseline features-based classifiers. The asymmetric formulation, on the other hand, succeeded at improving classification by assigning higher weights to spammers connections. This begs the question of whether an alternative and more expressive graphical model can be better suited to the task of social spam detection in the presence of weak prior beliefs. We discuss improvements to the current model in the next chapter.

Chapter 5

Spammers Detection with Conditional Random Fields

Chapter Contents

5.1 Introduction . . . 97
5.1.1 CRF vs. MRF . . . 98
5.1.2 Learning UGM parameters . . . 99
5.2 A CRF framework for Social Spammers Detection . . . 100
5.2.1 Problem Formulation . . . 101
5.2.2 Proposed solution . . . 101
5.3 Inference and learning in Conditional Random Fields . . . 103
5.3.1 Definition of Undirected Graphical Models . . . 103
5.3.2 Parameters Learning in Undirected Graphical Models . . . 105
5.3.3 Loss functions . . . 105
5.3.4 Inference . . . 107
5.4 Experimental Evaluation on Twitter . . . 109
5.4.1 Graphical Models Implementation . . . 109
5.4.2 Results and Discussion . . . 109
5.5 Conclusion . . . 112

5.1 Introduction

In the previous chapter, we have shown that Undirected Graphical Models (UGMs), particularly the Markov Random Field, offer a way to model uncertainty (biased priors) and dependency (similar users) in the context of social spammers detection. This chapter reiterates on the same research problem and experimental setting, with two main differences:

- The conditional random field (CRF), which is a discriminant graphical model, replaces the Markov Random Field (MRF) as a non-discriminant generative model.
- Parameters are optimized in an adapted learning framework that uses LBGFS for objective optimization, the tree reweighted (TRW) message passing algorithm for approximate inference and an Empirical Risk Minimization (ERM) objective loss function.

Before motivating these two changes, we start by broadly summarizing the components of the classification system proposed in the previous chapter, namely the potentials functions, the inference algorithm and the optimization (learning) method.

- Potentials:
  - Edge potential: A 2×2 matrix that associates a parameter to each of the four possible edge states.
  - Node potential: A non-parametric function that is exactly equal to the prior prediction computed on each account.
- Inference: The widely used LBP algorithm that approximates a marginal joint probability of the variables. The algorithm uses the edge potentials matrix as its propagation/update matrix, and the node potential as its initial configuration of states.
- Optimization: A 2-dimensional grid search of the best parameters of the edge potential. No optimization with respect to node potentials.

We will now explain how each of the proposed changes fills a gap in the above model.

5.1.1 CRF vs. MRF

MRF is a generative model typically used for learning a distribution over the states of nodes (output or labels in ML literature). It is often introduced in the context of a physical system such as the *Ising model*[1]. The machine learning literature, on the other hand, has used the conditional form of MRF, the conditional random field (CRF) in applications such as natural language processing [122] and computer vision [123].

Unlike MRF, CRF is a discriminative model that learns the output distribution conditioned on the input. This is especially important in the domain of social spam detection where correctly sampling the joint input/output distribution $p(x, y)$ is practically impossible due to the selection bias caused by collection and labeling methods [124]. This may imply that the CRF model is more robust to data bias and that its results can be more readily generalizable. Moreover, this formulation is more

[1]The Ising model has been used in statistical mechanics to model atomic spins that can take one of two states ($+1$ and -1). Spins graphs are usually modeled as lattices where each spin interacts with its direct neighbors. Extending this lattice model to a CRF framework has been useful in domains such as image processing.

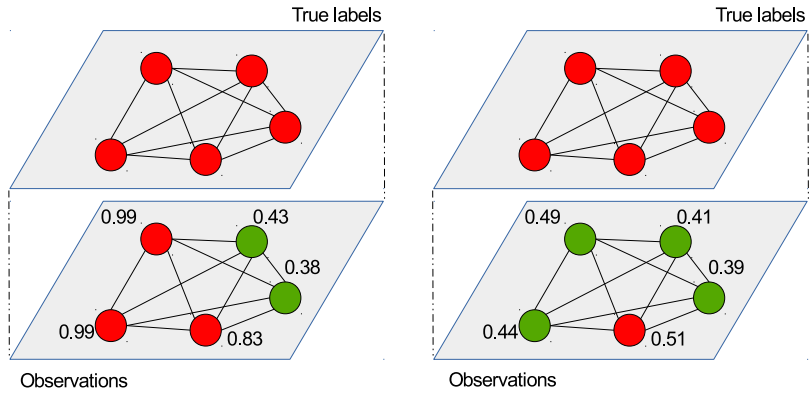

Figure 5.1: An example of two graphs having the same labels and different observations.

in line with the standard definition of a prediction problem where the goal is to learn the conditional distribution $p(y/x)$ for input x and output y. The CRF model is also capable of incorporating observations on nodes and edges. In particular, it makes it possible to model the bias in the prediction priors observed on each node by making both nodes and edges potentials dependent on prior beliefs.

Consider for example the two graphs in Figure 5.1. Both graphs have identical *labels* but different *observations* (in this case, different prediction priors computed over nodes). In the setting summarized above, learning would often map to optimizing the parameters of the edge potential matrix with respect to the negative log-likelihood of the joint configuration of *labels*. The parameters therefore depend solely on the labels distribution in the training dataset. In the case of Figure 5.1, this means that both examples would result in the same MRF model despite the difference of observations on the nodes. This discussion shows that the MRF model has a limited representative power that hinders the full exploitation of statistical features (or observations) on the defined random variables.

To the best of our knowledge, the MRF model is the only probabilistic graphical model that has been explored in the context of fraud, spam or abuse detection on online platforms including online social networks (e.g. Polonium [103] for malware detection, SpEagle for opinion fraud [106]). We are not aware of any contribution that proposes to use the CRF.

5.1.2 Learning UGM parameters

Elements of the setting summarized above and used in the previous chapter are common to many web-related applications. A drastic simplification would be to omit the MRF notation, and build the whole model around an update matrix and the Loopy Belief Propagation algorithm. This is applicable for both labels and beliefs propaga-

tion. Another more popular setting [106] that we used in Section 5.1.1 as one of two possible design choices, is to:

- Define an MRF.
- Set prior predictions as node priors.
- Propose one symmetric update matrix (that is usually skewed toward favoring the identical states on the extremities of an edge).
- Evaluate the performance of the model by computing posterior predictions using the above update matrix and the LBP algorithm.

In the previous chapter, we proposed to upgrade this architecture by allowing a free form update matrix and by evaluating a wide range of values on the parameters grid. Since the grid search was constrained, it was possible to avoid some of the pitfalls of the LBP algorithm that are associated with high coupling parameters. This advantage is lost when LBP is used as a part of an internal inference routine inside a gradient-based optimization algorithm.

In this framework, which has been used in previous works in neighboring domains [106], the MRF inference problem is predominantly solved with the Loopy Belief Propagation (LBP) algorithm, which iteratively computes posterior prediction probabilities for the defined variables. The parameters of this matrix are often set in an empirical way to favorize models that link identical classes.

Unfortunately, when learning the graphical model parameters from data, as we did in the previous chapter, using LBP for approximate inference can become problematic. In particular, this is true when edge weights are high in pairwise models [125]. This leads to the underestimation of the partition function, which in turn results overestimated marginal probabilities. To surmount this problem, we use tree reweighted message passing algorithm (TRW) for inference [126]. We also substitute the often-used negative log-likelihood loss function (NLL), with the clique loss (CL) [127]. By adopting empirical risk minimization rather than maximum likelihood estimation, the loss function is more aligned with the goal of minimizing classification errors.

Chapter Structure. The remaining of this chapter is structured as follows. Section 5.2 introduces the CRF framework that replaces the MRF framework as a solution to the social spammers detection problem. We discuss learning and inference in undirected graphical models in Section 5.3. Section 5.4 presents and discusses the experimental results of applying the CRF model to the Twitter dataset. We conclude the chapter in Section 5.5.

5.2 A CRF framework for Social Spammers Detection

In this section, we reiterate the problem of social spammers detection and introduce the notation and potentials associated with a CRF-based solution.

5.2.1 Problem Formulation

Setting. We reuse here the same problem setting we formulated in Chapter 4. We define namely:

- A set of users U.
- For each user $u \in U$:
  - A numerical features vectors $\boldsymbol{x}_u \in X$ for each $u \in U$.
  - A set of messages $M(u)$ posted by the user u.
  - A random variable Z_u designating the class of user u ($Z_u \in L$ and $L = \{0, 1\}$).
  - A prediction $p(z_u = 1)$ of a user's class based on its features vector $\boldsymbol{x}_u$.

We also have the same assumption stating that users having content in common are more likely to have the same class and define the same content-based similarity over users.
Goal. The goal is to build a classifier that, for each user u, predicts a class $Z_u \in L$. The classifier should enhance the prior predictions by incorporating dependencies between users as captured by the defined graph of similarity.

5.2.2 Proposed solution

Instead of the MRF proposed in chapter 4, we propose in this chapter to use the pairwise CRF. Unlike MRF, which can be viewed as a generalization of Naive Bayes, CRF can be viewed as a generalization of logistic regression. The contrast between the two models is therefore fundamentally one of modeling the joint input distribution versus modeling the output distribution conditioned on the input. For input x and output y this translates to MRF modeling $p(x, y)$ and CRF modeling $p(y/x)$.

The general architecture of the overall system is similar to the one we previously proposed. The system's data flow can be seen in Figure 5.2 and has the main three parts introduced for the MRF solution, mainly: priors generation, graph construction and computing CRF predictions. Since the first two have been discussed in chapter 4, we focus here on the last part. We discuss hereafter the forms of the CRF model potentials.

In the following, we define node and edge potentials of the CRF model. Local beliefs offer a valuable indication of whether the account is a spammer, while similarity offers a way to propagate beliefs on the network of users. Having the right balance between node and edge potentials helps to effectively exploit these two aspects of the problems.

5.2.2.1 Node Potentials

Node potentials are factors, positive numbers that quantify the potential of a node of being in a given state. Here states are defined by the classes the user can belong to, namely legitimate and spammer. In the CRF model, node potentials $\phi_u(z_v, \boldsymbol{y}; \boldsymbol{\theta})$

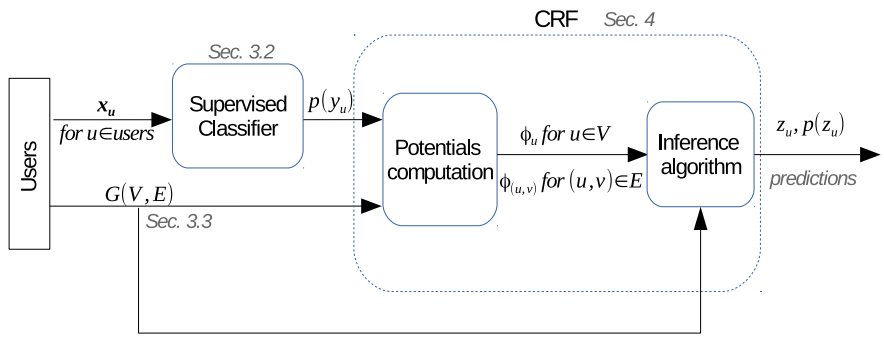

Figure 5.2: General architecture of the proposed system.

Table 5.1: The edge potential matrix $\phi_{(u,v)}(z_u, z_v)$.

		z_v	
		Legitimate	Spammer
z_u	Legitimate	e^{θ_4}	e^{θ_6}
	Spammer	e^{θ_5}	e^{θ_7}

can depend of both the state z_v of node v and the local features describing the node $\boldsymbol{y}$. They incorporate prior beliefs on nodes classes and are chosen to drive the UGM prediction to match the prior belief on a node. In this case, the expression of the node potential is given in equation 5.1:

$$\phi_u(z_v, \boldsymbol{y}; \boldsymbol{\theta}) = \begin{cases} \exp(\theta_0.y_0 + \theta_2.y_1) & \text{if } z_v = 0 \text{ (legitimate user)} \\ \exp(\theta_1.y_0 + \theta_3.y_1) & \text{if } z_v = 1 \text{ (spammer)} \end{cases} \tag{5.1}$$

where z_v is the state (class) of node v and $\boldsymbol{y} = [y_0, y_1]$ is the features vector defined by $y_0 = 1$ and $y_1 = p(\text{user v is a spammer})$ is the prior belief that a user is a spammer.

In general, the aim of a node potential is to ensure that the class predicted for each node is in line with the observations (features) associated with the node.

5.2.2.2 Edge Potentials

An edge (or pairwise) potential is defined on each edge in the graph E. The edge potential $\phi_{(u,v)}$ is defined as a matrix of parameters ($\exp(\theta_i)$ for $\theta_i \in \mathbb{R}$) which associates a positive real number to the possible combinations of (z_u, z_v) (see the edge potential matrix in Table 5.1). Negative values of θ_i result in smaller potentials and are associated with a repulsive edge. Positive values of θ_i are associated with attractive edges.

In models where edges connote a notion of similarity between nodes, it is generally desirable for the edges to be attractive between nodes having the same state

and repulsive between nodes of opposing states. This ensures smoothness between connected nodes. In the current context, edge potentials are chosen such that the value of the potential between identical classes (e.g. between two spammers or between two legitimate accounts) are stronger than the potential value between different classes (one spammer and one legitimate account). Note that since edges in the current application stand for only one similarity concept, only one edge potential matrix is defined.

5.3 Inference and learning in Conditional Random Fields

In this section, we discuss learning and inference in probabilistic graphical models, with a special focus on conditional random fields. We start by introducing common notation and restrict the discussion to the universally used pairwise log-linear model (Section 5.3.1). We then introduce the conditional random field model and underline the notation changes needed to apply general learning and inference to the CRF model. We follow this by discussing parameters learning (Section 5.3.2), and motivate our choices for inference methods (Section 5.3.4) and loss functions (Section 5.3.3).

5.3.1 Definition of Undirected Graphical Models

Undirected graphical models (UGM) is a general name for a family of models that include Markov Random fields (MRF) and their conditional variation, the conditional random fields (CRF). They are often referred to simply as Markov random fields, and the notation, as we will see afterwards, makes it possible to discuss learning and inference in all undirected models using the Markov Random Fields model. These undirected models are used as a compact and powerful tool to model dependencies between random variables under Markov assumptions [128]. The Markov Assumption is the property of a model where a (future) state of a given process depends only on the (present) state of this process. This is, for instance, the case in a hidden Markov model. Note the use of temporal terms to denote a temporal or sequential relationship between states in a hidden Markov model. In MRF, this property is extended to a randomly connected graph where the dependency between two nodes is not sequential. A notable example of this in the is the Ising model, where the MRF is defined over a two-dimensional lattice with a periodically repeated structure.

Definition. An MRF (G, Ψ) is defined by a set of variables V, a graph $G(V, E)$ coding dependencies between these variables, and a set of potential functions Ψ. The notion of a clique[2] is of a central importance in the MRF framework. An MRF is said to be factorized over the cliques of a graph G if the joint probability distribution $p(\boldsymbol{Z} = \boldsymbol{z})$ can be written as the product of potentials Ψ defined over cliques in the graph G as follows:

$$p(\boldsymbol{Z}) \propto \prod_{C \in cl(G)} \Psi_C(\boldsymbol{Z}), \tag{5.2}$$

[2]A clique $C \subset G(V, E)$ of a graph G is a fully-connected subset of its vertex set V, i.e., $(u, v) \in E$ for all $u, v \in C$.

where $cl(G)$ is the set of cliques of G. Note that as per the definition above, potentials (or factors) need to be positive but they are not necessarily probability distributions over the input space.

Pairwise models. The simplest and most used form of MRF is the pairwise model, where factorization is limited to cliques of the second order. In our case, $V = \{Z_u\}_{u \in \text{Users}}$ is the set of random variables corresponding to users, and $E = \{(u, v)\}$ is the set of edges in the users graph. Potentials can be thus defined over nodes (called unary potentials and denoted ϕ_u) and edges (called pairwise potentials and denoted $\phi_{u,v}$). The probability of a configuration Z over all random variables in a pairwise model is defined as:

$$p(\boldsymbol{Z}) = \frac{1}{\Xi} \prod_{u \in V} \phi_u(Z_u) \prod_{(u,v) \in E} \phi_{u,v}(Z_u, Z_v), \tag{5.3}$$

where Z_u and Z_v are the random variables denoting the classes of users u and v respectively, and Ξ is the partition function[3], defined as:

$$\Xi = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\boldsymbol{z}),$$

where $\mathcal{Z}$, the space of all possible configurations, is the domain of $\boldsymbol{z}$, the output vector.

Log-linear models. Since the factors above are positive functions, a factorized MRF can be represented as an exponential family. This representation facilitates the expression of learning and inference in the upcoming sections.

Let $\boldsymbol{\theta}$ be the vector of CRF parameters defining the node and edge potentials ϕ_u and $\phi_{u,v}$, and consider that potentials ϕ_u and $\phi_{u,v}$ are defined in terms of log-potentials as linear functions of the parameters $\boldsymbol{\theta}$. The joint probability defined in (5.3) can be written as:

$$p(\boldsymbol{z}; \boldsymbol{\theta}) = \frac{1}{\Xi(\boldsymbol{\theta})} \exp\left(\boldsymbol{\theta} \cdot \mathbf{f}(\boldsymbol{z})\right), \tag{5.4}$$

where $\mathbf{f}(z)$ is the vector of sufficient statistics defined as:

$$f(Z) = \{I[Z_c = z_c] \mid \forall c, z_c\} \cup \{I[Z_u = z_u] \mid \forall u, z_u\}. \tag{5.5}$$

And $\boldsymbol{\theta} \cdot \mathbf{f}(z)$ is the inner product defining the potentials as linear functions of the parameters $\boldsymbol{\theta}$.

Let $A(\boldsymbol{\theta})$ be the log-partition function, which will appear repeatedly in the learning and inference sections. The log-partition function is defined as:

$$A(\boldsymbol{\theta}) = \log \Xi(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z} \in \mathcal{Z}} \exp \boldsymbol{\theta} \cdot \mathbf{f}(\boldsymbol{z}). \tag{5.6}$$

[3]The partition function is denoted as Z in the Machine Learning literature. We instead use the physics notation Ξ in this work to avoid confusion with Z the vector of random variables associated with posterior predictions.

The probability can be written as:

$$p(\boldsymbol{z};\boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta}\cdot\mathbf{f}(\boldsymbol{z})\right)}{\exp A(\boldsymbol{\theta})} = \exp\left(\boldsymbol{\theta}\cdot\mathbf{f}(\boldsymbol{z}) - A(\boldsymbol{\theta})\right). \tag{5.7}$$

Conditional Random Field. In this chapter, we focus on the conditional variant of MRF, the Conditional Random Field (CRF) [122] where the potentials are conditioned on the input features (not to be confused with sufficient statistics $\mathbf{f}$ defined above). Unlike MRF, CRF is a discriminant model that learns the distribution of labels given the input features rather than learning the general labels distribution. The probability of a configuration $\boldsymbol{z}$ given $\boldsymbol{y}$, namely the vector of input features, is defined as:

$$P(\boldsymbol{z} \mid \boldsymbol{y};\boldsymbol{\theta}) = \frac{1}{\Xi(\boldsymbol{y},\boldsymbol{\theta})}\prod_{u\in V}\phi_u(z_u,\boldsymbol{y};\boldsymbol{\theta})\prod_{(u,v)\in E}\phi_{u,v}(z_u,z_v,\boldsymbol{y};\boldsymbol{\theta}). \tag{5.8}$$

For simplicity, we use the general graphical models notation in the rest of this section. The corresponding CRF notation can be simply obtained by considering that $\boldsymbol{\theta}$, the free parameters vector, is a function of $\boldsymbol{y}$ and of free parameters.

5.3.2 Parameters Learning in Undirected Graphical Models

Learning the parameters $\boldsymbol{\theta}$ of undirected graphical models is defined as finding the model's parameters that minimize an objective loss function $L(\boldsymbol{\theta})$. The minimization is usually performed via unconstrained differentiable optimization algorithms such as LBGFS. These algorithms are variants of gradient-descent and involve repeated computation of the gradient of the loss function. As we show in the next section, computing the gradient of the loss typically requires estimation of marginal probabilities, which is done by performing inference at each iteration. We thus start by discussing loss functions commonly used in the context of learning in probabilistic graphical models and then discuss approximate inference methods that have been proposed in the literature to estimate marginal probabilities.

5.3.3 Loss functions

A loss function is a way to quantify how well a model fits the data. The goal of learning is to find the model's parameters values that minimize the defined loss. While loss has been traditionally defined in terms of Maximum Likelihood Estimation (MLE), it is increasingly common to have Empirical Risk Minimization (ERM) guide the choice of the loss function. We discuss these two concepts below.

5.3.3.1 MLE and the Log-Likeliohood Loss

The negative log-likelihood (NLL) loss, based on the concept of Maximum Likelihood Estimation is the most widely known loss. In Bayesian terms, the minimization of

NLL is equivalent to finding a MAP estimation of the model parameters. For a graphical model with a joint variables configuration $\boldsymbol{z}$ and a parameters vector $\boldsymbol{\theta}$, NLL is defined as follows:

$$NLL(\boldsymbol{\theta}, \boldsymbol{z}) = -\log p(\boldsymbol{z}; \boldsymbol{\theta}) = A(\boldsymbol{\theta}) - \boldsymbol{\theta} \cdot \boldsymbol{f}(z). \tag{5.9}$$

It follows that the derivative of the loss is:

$$\frac{dL}{d\boldsymbol{\theta}} = \boldsymbol{f}(z) - \boldsymbol{\mu}(\boldsymbol{\theta}), \tag{5.10}$$

where $\boldsymbol{\mu}$, the mean-parameters is defined as:

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \frac{dA(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \sum_{\boldsymbol{z}} p(\boldsymbol{z}; \boldsymbol{\theta}) f(\boldsymbol{z}). \tag{5.11}$$

Equation 5.11 shows that computing the gradient requires computing the marginal probabilities $p(\boldsymbol{z}; \boldsymbol{\theta})$, which is generally intractable in graphical models. The log-likelihood loss is thus often replaced by a (surrogate) likelihood that uses approximate probabilities computed by approximate inference methods discussed in Section 5.3.4).

5.3.3.2 ERM and the Clique Loss

In most machine learning applications (such as image processing, computer vision and object tracking), the model used to represent the data is often mis-specified, meaning that it is a simplification of the observed phenomenon. Since maximum likelihood learning is adapted for well-specified models, it is increasingly replaced by empirical risk minimization [129, 130]. This is simply motivated by the observation that, in classification problems, we are more interested in minimizing the number of classification errors (empirical risk minimization) than maximizing the joint probability (maximum likelihood estimation). And thus, it is reasonable for the loss to be tied to the marginal distribution of each variable rather than the full joint distribution. The loss is defined as follows:

$$R(\boldsymbol{\theta}) = \sum_{\hat{z}_u} L(\boldsymbol{\theta}, \hat{z}_u), \tag{5.12}$$

where the sum is done over $\hat{z}_u$ the instances of variables in the training data.

In this work, we use a loss defined on cliques instead of the univariate loss used in 5.12. This has the advantage of being consistent since univariate loss can produce inaccurate joint probabilities despite correctly predicting marginal probabilities. Let c denote a clique of the graph $G(V, E)$. Then, the Clique Loss is defined over all the cliques (edges in the case of pairwise models) as:

$$L(\boldsymbol{\theta}, \boldsymbol{z}) = -\sum_{c} \log \boldsymbol{\mu}(\boldsymbol{z}_c; \boldsymbol{\theta}), \tag{5.13}$$

where $\boldsymbol{\mu}(\boldsymbol{z}_c; \boldsymbol{\theta})$ is defined similarly to $\boldsymbol{\mu}(\boldsymbol{z}; \boldsymbol{\theta})$ by using indicator functions on cliques:

$$\boldsymbol{\mu}(\boldsymbol{z}_c; \boldsymbol{\theta}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{X}, \boldsymbol{\theta}) I[\boldsymbol{z}'_c = \boldsymbol{z}_c] = p(\boldsymbol{z}_c; \boldsymbol{\theta}). \tag{5.14}$$

5.3.4 Inference

We have shown in the last section that learning requires the estimation of marginal distributions. This is known as *inference* in the context of graphical models. From 5.7, it follows that computing marginals requires computing the partition function. This is intractable because it consists of summing the joint probability over all possible labels permutations.

While efficient inference algorithms have been designed for trees and chains, they are not applicable to general graphs with cycles. The approach to compute approximate marginals is to express the log-partition function using the exact variational principle, thus recasting the computation problem as an optimization problem that can be then approximated. We start by defining the exact variational principle before introducing these two algorithms for approximate marginal inference in general undirected graphical models: Loopy belief propagation (LBP) and tree-reweighted message passing (TRW). Both are message passing algorithms that produce approximate marginals.

5.3.4.1 Exact Variational Principle

According to the exact variational principle, the partition function can be defined as the minimum of the true (Gibbs) free energy over the marginal polytope [131, 132]. The log-partition function can be represented as:

$$A(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu}), \tag{5.15}$$

where:

- $\mathcal{M}$ is the marginal polytope defined as the set of realizable mean parameters, namely
$$\mathcal{M} = \{\boldsymbol{\mu}' : \exists \boldsymbol{\theta}, \boldsymbol{\mu}' = \boldsymbol{\mu}(\boldsymbol{\theta})\}$$
- and $H(\boldsymbol{\mu})$ is the entropy defined as:
$$H(\boldsymbol{\mu}) = -\sum_{z} p(z; \boldsymbol{\theta}(\boldsymbol{\mu})) \log p(z; \boldsymbol{\theta}(\boldsymbol{\mu})).$$

Unfortunately, in general graphs, computing the entropy is intractable and the marginal polytope is difficult to characterize [127]. The approximate marginal inference algorithms that we will discuss below can be seen as minimizing a relaxation of the exact variational principle.

5.3.4.2 Loopy Belief Propagation

Belief propagation (BP) [133] is a sum-product message passing algorithm that is widely applied to marginal inference in graphical models. For MAP marginal inference (finding the configuration that minimizes the energy), the max-product BP is used to obtain beliefs on marginal distributions [134].

The loopy belief propagation (LBP) algorithm [119] provides an extended variant of the standard BP that can be used for approximate inference in graphs with a general structure, including graphs with cycles (e.g. in Computer Vision applications [120]). While the uniqueness of the LBP solution is only guaranteed for trees and single-cycle graphs, the algorithm may have multiple fixed points in general. As a consequence, LBP has no convergence guarantees over graphs with cycles. In practice, however, it is known to give good results and often converges within few iterations.

In terms of computational complexity, the LBP algorithm is considered linear in the number of edges. Its computational complexity is $\mathcal{O}(d\,|E|)$, where d is the number of iterations required until convergence and $|E|$ is the number of edges.

5.3.4.3 Bethe free energy minimization

The LBP algorithm has tight connections with the Bethe free energy minimization principle in statistical physics [135]. It turns out that the fixed points obtained from the LBP algorithm are the local minima of the Bethe free energy, which is defined in the context of variational inference to approximate the partition function. The Bethe approximation introduces two changes to the exact variational equation (5.15): the true entropy is replaced by the Bethe entropy, and the marginal polytope is relaxed to the local pseudo-marginal polytope (where only pairwise consistency is required in the marginal vector) [136].

In some cases, Bethe approximation has been shown to be inaccurate [125]. This happens when coupling (namely, the strength of edge potentials weights) is high in pairwise models. In our case, this probably happens because the configuration ($x_u = x_v = 1$) tends to be associated with high values of $\boldsymbol{\theta}$, especially when the learning algorithm explores higher values of the parameter. As a consequence, the underestimation of the partition function results in inaccurate estimation of marginals (with the probabilities of certain configurations becoming larger than 1).

5.3.4.4 Tree-Reweighted Message Passing

Due to the drawbacks associated with LBP and Bethe approximation, which we encountered consistently during the parameters learning of the CRF model in our particular experimental setting (see Section 5.4), we will use the tree-reweighted message passing (TRW) algorithm for marginal inference instead of LBP.

Tree-reweighted message passing (TRW) was first defined in [137] as a max-product message passing algorithm for energy minimization. The sequential-TRW (S-TRW) was then proposed in [126] as a provably convergent algorithm. On certain problems, TRW has been shown to yield lower energy than both belief propagation and graph-cuts [126, 138]. TRW is based on the idea of maximizing a convex lower bound on the energy by solving the MAP problem on a convex combination of trees derived from the original general graph. The optimization on trees is performed using the BP algorithm which is convergent on trees. A distribution ρ that

conserves the energy over the original graph is defined over the trees collection and a reparametrization of messages is performed according to ρ.

In the variational perspective, TRW replaces the local polytope $\mathcal{M}$ with a superset $\mathcal{L} \supset \mathcal{M}$. The associated variational equation approximates an upper bound to the log-partition function:

$$\tilde{A}(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{L}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + \tilde{H}(\boldsymbol{\mu}), \tag{5.16}$$

where $\tilde{H}(\boldsymbol{\mu})$ is the so-called TRW entropy [137].

5.4 Experimental Evaluation on Twitter

We present in this section the results of applying the proposed CRF model to the Twitter dataset introduced in chapter 3. We also reuse the same experimental setting described in chapter 4 for:

- Graph construction.
- Prior predictions generation.

We discuss parameters choice and the efficiency of the optimization algorithm used to learn CRF parameters.

5.4.1 Graphical Models Implementation

CRF is implemented on top of the JGMT package [127] and inference is performed via the TRW algorithm. JGMT, a package originally conceived for image processing applications, is written in Matlab with the core functionalities written in C++. Unlike other existing CRF libraries which only allow chain models, JGMT allows the definition of models with a general graph structure and implements the TRW algorithm[4].

For CRF parameters learning, we use the MinFunc optimization package [139] in Matlab with the default L-BFGS function and define the objective loss as the Clique Loss in JGMT. Since the loss is not convex, we launch the optimization 100 times starting from random initial parameters sampled from the normal distribution. This is done for each possible combination of experimental settings (features set and classification model).

5.4.2 Results and Discussion

Table 5.2 compares the classification performance of baseline supervised classifiers, the MRF models (with both the symmetric and asymmetric configurations introduced in chapter 4) and the CRF model. The performance is measured in terms of recall, precision and the F1-measure.

[4]We used the *UGM* library [121] in Matlab for our MRF-based model (chapter 3). Although it also supports CRF, *UGM* uses LBP for inference and does not implement TRW.

Table 5.2: Classification performance of local supervised classifiers, the symmetric and asymmetric MRF models and the CRF-based model.

		All features				**Benevenuto features**				**Stringhini features**			
		Sup.	Sym. MRF	Asym. MRF	CRF	Sup.	Sym. MRF	Asym. MRF	CRF	Sup.	Sym. MRF	Asym. MRF	CRF
SVM	Precision	0.891	0.917	**0.919**	0.909	0.939	**0.955**	0.930	0.945	0.941	**1.000**	0.966	0.980
	Recall	0.598	0.571	0.883	**0.909**	0.756	0.273	0.857	**0.896**	0.195	0.143	0.364	**0.636**
	F1	0.715	0.704	0.901	**0.909**	0.838	0.424	0.892	**0.920**	0.323	0.250	0.528	**0.772**
	Accuracy	0.752	0.764	0.904	**0.911**	0.847	0.637	0.898	**0.924**	0.573	0.580	0.682	**0.815**
LR L1	Precision	0.865	0.86	0.890	**0.924**	0.961	**1.000**	0.924	0.969	**1.000**	**1.000**	**1.000**	**1.000**
	Recall	0.549	0.558	0.844	**0.948**	0.598	0.545	0.792	**0.805**	0.159	0.325	0.506	**0.623**
	F1	0.672	0.677	0.867	**0.936**	0.737	0.706	0.853	**0.879**	0.274	0.490	0.672	**0.768**
	Accuracy	0.720	0.739	0.873	**0.936**	0.777	0.777	0.866	**0.892**	0.561	0.669	0.758	**0.815**
LR L2	Precision	0.956	0.900	0.933	**0.958**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Recall	0.524	0.468	0.727	**0.883**	0.317	0.182	0.714	**0.779**	0.159	0.325	0.481	**0.558**
	F1	0.677	0.615	0.818	**0.919**	0.481	0.308	0.833	**0.876**	0.274	0.490	0.649	**0.717**
	Accuracy	0.739	0.713	0.841	**0.924**	0.643	0.599	0.86	**0.892**	0.561	0.669	0.745	**0.783**
RF	Precision	**0.955**	0.924	0.902	0.939	**1.000**	0.933	0.928	0.923	0.960	**1.000**	0.925	0.972
	Recall	0.780	0.792	0.961	**1.000**	0.585	0.727	0.831	**0.935**	0.585	0.532	0.805	**0.896**
	F1	0.859	0.853	0.931	**0.969**	0.738	0.818	0.877	**0.929**	0.727	0.695	0.861	**0.932**
	Accuracy	0.866	0.866	0.930	**0.968**	0.783	0.841	0.885	**0.930**	0.771	0.771	0.873	**0.936**

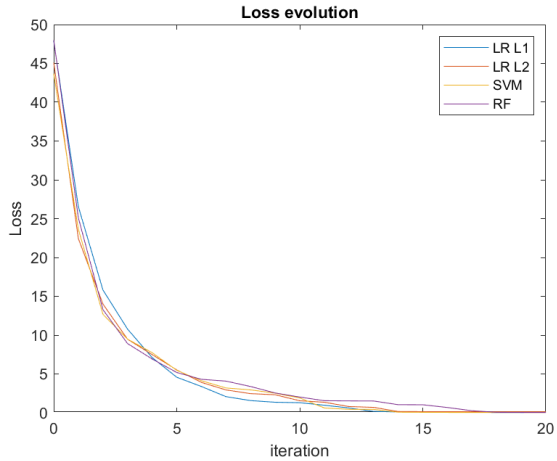

Figure 5.3: The evolution of loss as a function of the iterations of the optimization algorithm (average over 100 runs).

Compared to the baseline beliefs, CRF consistently improves recall by a range of 11%-46% with an average of 31% while simultaneously maintaining the original precision achieved by local classifiers. The CRF also consistently outperforms both MRF models.

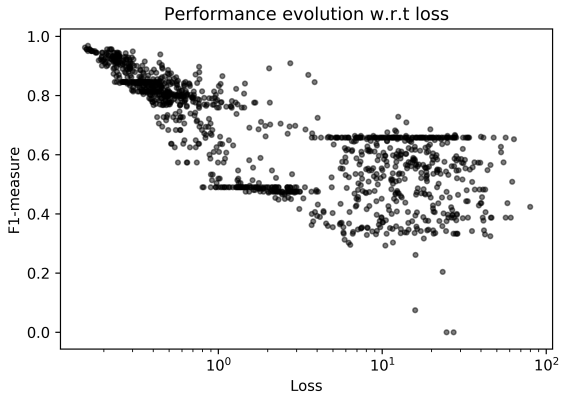

Figure 5.4: The F1-measure as a function of loss for the 100 runs of the learning algorithm.

5.4.2.1 Efficiency

The algorithm used to learn CRF parameters converges on average in 1.5 seconds. Figure 5.3 shows the evolution of the loss as a function of iterations of the learning algorithm.

5.4.2.2 Objective Loss

Figure 5.4 shows the relationship between the Clique Loss and the F1-measure which we use as a proxy for the classification performance. Each point in the plot corresponds to a local optimum the algorithm has converged to for one of the 100 runs. The F1-measure is computed on the predictions associated with the parameters the algorithm has converged to. For small losses ($L < 2$), there is a high correlation between the loss and the F1-measure with a correlation coefficient of -0.82 on average).

5.4.2.3 CRF parameters

We compare here the parameters learned for CRF and MRF models.

Node potentials. The parameters of the node potentials determine how the graphical model exploits the information conveyed by the prior belief on each node (user). The CRF formulation of node potentials results in flatter curves compared to the exponential curve of the MRF model as shown in Figure 5.5. The CRF curve has an attenuating effect over prior belief. This effect is most pronounced on the extreme values of the belief.

- Case 1: $p(x = 1) \rightarrow 0$ (legitimate label predicted with high confidence). The potentials ratio curve assigns a non-zero potential for the state $x = 1$ even

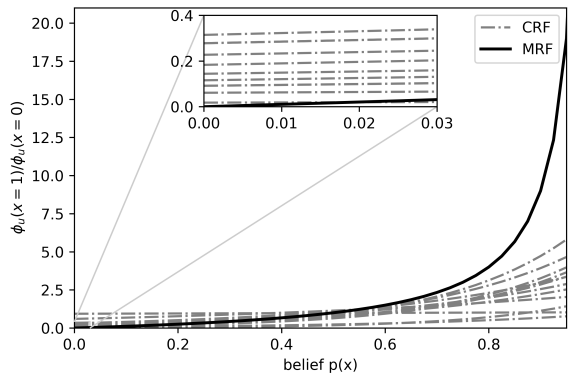

Figure 5.5: The ratio $\phi_u(x = 1)/\phi_u(x = 0)$ of the node potential for the two possible states as a function of the node belief $y_1 = p(x = 1)$ for beliefs computed using the different possible configurations of supervised classification models.

when the belief $p(x)$ computed by the local node classifier is zero. This has the effect of allowing the model to predict that a node is a spammer even when the local classifier is very confident of the legitimate prediction.

- Case 2: $p(x = 1) \rightarrow 1$ (spammer label predicted with high confidence). The potentials ratio curve for MRF also follows a steep curve when the prior belief approaches 1. CRF attenuates this effect, allowing the model to predict a user as legitimate even when the local classifier is very confident that a user is a spammer.

5.5 Conclusion

In this chapter, we proposed to use the Conditional Random Field to tackle the problem of detecting spammers on Online Social Networks. Like MRF, the proposed framework allowed including elements from both the supervised and the graph-based detection paradigms. But unlike MRF, the CRF framework enabled conditional learning of the output distribution based on the input features. In this case, node potentials were expressed as functions of prediction priors. This resulted in a model that is more expressive and more robust than traditional non-conditional probabilistic models.

We also discussed learning and inference in undirected graphical models and motivated our choice for the TRW algorithm as an inference algorithm and the clique loss as an ERM loss. We showed the effectiveness of the proposed model by evaluating it on the Twitter dataset. In addition to offering more robustness, results showed that the CRF model has better classification performance than the MRF model proposed in the previous chapter.

Chapter 6

Conclusion

For online social networks such as Facebook and Twitter, controlling social spam is of the upmost importance. Failing to do so would negatively impact users experience and safety. More importantly, it would mean that these platforms will continue to allow mass manipulation, political propaganda and interference with the free information flow.

The supervised detection paradigm has dominated the social spam detection scene for a long time. This is quite understandable since a supervised pipeline is elegantly simple. Its simplicity makes it the first design choice in a lot of machine learning applications. But like any system deployed in a dynamic environment, the performance of a supervised detection system is not constant. On a shopping website, variation in buying trends may engender a variation in classification performance. In the case of social spam detection, this is rendered even more complex by the adversary nature of the target.

This work aimed at proposing a solution to the problem of the degrading performance of supervised classifiers in the face of the dynamic task of detecting social spammers. Departing from the assumption that the modus operandi of social spammers makes their content inherently linked, we proposed to model the problem using a probabilistic graphical framework. This allows to model the detection task as a joint classification over connected social accounts. In this probabilistic framework, statistical features of the accounts are used to generate prior predictions probabilities. And belief can then be propagated and updated over connected users. We showed that, using an adequate similarity measure, it is possible to reach posterior beliefs that are closer to the observed labels than what is previously predicted by features-based classifiers.

This work represented a step towards the goal of leveraging accounts dependency for exploiting scarce, inaccurate and biased predictions from variable sources (including weak statistical classifiers) and towards building detection systems that are more robust to features variations.

Let us now systematically examine the research objectives laid out in the introduction of the manuscript.

Contribution 1: Evaluating and characterizing the effect of spam evolution on the classification performance of state-of-the-art supervised systems.

In Chapter 3, we have evaluated a large range of statistical detection features proposed in the literature on a dataset of labeled Twitter accounts. These features were issued from state-of-the-art systems that reported excellent detection performance in their original papers. The results of evaluating these systems on a recent population of spammers using multiple machine learning supervised classifiers, showed a clear deterioration in the classification performance of these features. This shows that these state-of-the-art features are insufficient to characterize the current population of spam accounts on Twitter, and that they are thus not a generalizable detection tool.

The classification task at hand was severely unbalanced with estimates placing legitimate users between 85% and 95% of the total Twitter population. In such settings, a random classifier assigning labels according to the labels distribution can achieve high accuracy. The classification results obtained in Chapter 3 confirmed that features-based classifiers outperform a random classifier or a majority classifier. These classifiers were thus weak but better than a random baseline.

Results also suggest that the deterioration in performance was not random and that some spammers are more affected than others. By further studying the distribution of prediction probabilities of features-based classifiers, we found that legitimate users were generally assigned a very low probability of being spammers, while the prediction probabilities of spammers had a wider distribution.

These observations led us to postulate that, by exploiting the confident predictions of these classifiers, they can be used for discovering spammers. We proposed in particular to use prediction probabilities as an input for a probabilistic model instead of binary predictions.

Contribution 2: Proposing a definition of similarity that effectively captures spammers collusion and ensures a high degree of class homophily.

Unsupervised approaches are built on the assumption that an abnormally high level of collusion is an indicator of an abusive behavior. Various platforms such as Facebook and YouTube have proposed graph construction mechanisms that allow detection of collusive behavior. We are not aware of a similar work on Twitter. Prior graph-based approaches on Twitter have been based on the social graph and an assumption of connections of *trust*. More in line with an assumption of behavioral collusion, we propose a novel definition of similarity based on a bipartite user-content interaction graph instead of the commonly used social graph.

Unlike unsupervised systems, we performed no thresholding on dense connections and are thus able to apply our system on smaller users clusters (both spammers clusters and legitimate clusters). By evaluating the proposed graph construction algorithm on the collected Twitter dataset, we also confirmed that content-based similarity implies indeed a high degree of class homophily.

Building a similarity graph is typically computationally intensive. It is exponential in the number of nodes. By emulating realistic Twitter settings, we have shown

that the proposed graph construction algorithm can be run reasonably fast on cloud commodity hardware.

Contribution 3: Proposing an undirected graphical models framework that exploits the proposed notions of belief and similarity.

We proposed in Chapter 4 a spam detection system based on probabilistic graphical models. The classification task, usually framed over individual accounts, became a joint optimization over connected accounts. Prediction probabilities of features-based classifiers were represented as node priors, and class homophily was modeled using edge potentials.

The proposed approach is a hybrid approach that allowed to include traditional features-based classifiers in a graphical formulation.

Contribution 4: Assessing the effectiveness of undirected graphical models in mitigating the effects of spam evolution on the performance of supervised classifiers.

MRF and CRF models both allowed including elements from the supervised and the graph-based detection paradigms. MRF offered a generative model that is more prone to generalizing the edge distribution of the training samples. CRF enabled conditional learning of the output distribution based on the input features. When evaluated on the Twitter dataset, CRF slightly outperformed MRF. Moreover, it has the advantage of being more expressive and more robust to input distribution variation than traditional MRF models. It is therefore expected to generalize better than the MRF model.

Results showed that traditional account-based supervised models, despite being inaccurate and lacking graceful degradation, can be effectively exploited in the context of the proposed probabilistic framework. Compared to individual prior predictions, the probabilistic formulation led to a significant increase in recall while maintaining high precision. This validates that features-based classifiers, while not outperforming in a classification task, can be utilized for discovery of spam accounts with high probability and that uncovered spam accounts can effectively be used to *seed* classification systems by acting as prior predictions for belief propagation. The notion of a weak local classifier is therefore effectively exploitable in the context of probabilistic graphical inference.

Perspectives and Future Work

We have shown in this thesis that some characteristics of spammers behavior on OSNs (namely their collusive posting pattern) can be exploited as a stable component of a defense system. This characteristic, when integrated with weak statistical classifiers in a probabilistic framework, allows to increase the performance of a spammers detection system.

The observed gap between the baseline of the used classifiers and the hybrid system we proposed can serve as an indicator for spam evasion. An increasing gap

can thus indicate active evasion. Characterizing and monitoring evasion in real time is an interesting and useful research direction, one that we are not aware has been previously pursued.

In the same active monitoring spirit, an equally useful next step is to consider proactive defense systems. Proactivity refers to two aspects:

- Forecasting the change in spammers behavior and characteristics through simulations. The evasion monitoring discussed above would offer up to date input for a forecasting system.
- Rendering the static classification component of the system dynamic by periodically retraining the classifiers using the output of the proposed system.

This work has a limited experimental setting. Generalizing the implications of the obtained results to a full-fledged in-the-wild application would require a larger dataset and further validation.

Appendix A

Features extracted from Twitter accounts

This appendix presents the 145 attributes extracted from each account.

1. active_tweeting_frequency_per_day
2. adjusted_nb_of_uses_of_hashtag
3. adjusted_nb_of_uses_of_mention
4. adjusted_nb_of_uses_of_sources
5. adjusted_nb_of_uses_of_url
6. age
7. avg_intertweet_times
8. avg_intertweet_times_seconds
9. content_duration_days
10. date_newest_tweet
11. date_oldest_tweet
12. default_profile
13. default_profile_image
14. diversity_index_of_hashtags
15. diversity_index_of_mentions
16. diversity_index_of_sources
17. diversity_index_of_urls

18. favourites_count
19. followees_per_followers_sq
20. followers_count
21. followers_count_minus_2002
22. followers_per_followees
23. friends_count
24. friends_count_minus_2002
25. hashtags_used_on_average
26. lang
27. len_description
28. len_screen_name
29. max_intertweet_times
30. max_intertweet_times_seconds
31. max_nb_characters_per_tweet
32. max_nb_favourites_per_tweet
33. max_nb_hashtags_per_tweet
34. max_nb_hashtags_per_word_in_the_tweet
35. max_nb_mentions_per_tweet
36. max_nb_mentions_per_word_in_the_tweet
37. max_nb_retweets_per_tweet
38. max_nb_symbols_per_tweet
39. max_nb_symbols_per_word_in_the_tweet
40. max_nb_urls_per_tweet
41. max_nb_urls_per_word_in_the_tweet
42. max_nb_words_per_tweet
43. mean_nb_characters_per_tweet
44. mean_nb_favourites_per_tweet
45. mean_nb_hashtags_per_tweet

46. mean_nb_hashtags_per_word_in_the_tweet
47. mean_nb_mentions_per_tweet
48. mean_nb_mentions_per_word_in_the_tweet
49. mean_nb_retweets_per_tweet
50. mean_nb_symbols_per_tweet
51. mean_nb_symbols_per_word_in_the_tweet
52. mean_nb_urls_per_tweet
53. mean_nb_urls_per_word_in_the_tweet
54. mean_nb_words_per_tweet
55. median_nb_characters_per_tweet
56. median_nb_favourites_per_tweet
57. median_nb_hashtags_per_tweet
58. median_nb_hashtags_per_word_in_the_tweet
59. median_nb_mentions_per_tweet
60. median_nb_mentions_per_word_in_the_tweet
61. median_nb_retweets_per_tweet
62. median_nb_symbols_per_tweet
63. median_nb_symbols_per_word_in_the_tweet
64. median_nb_urls_per_tweet
65. median_nb_urls_per_word_in_the_tweet
66. median_nb_words_per_tweet
67. mentions_used_on_average
68. min_intertweet_times
69. min_intertweet_times_seconds
70. min_nb_characters_per_tweet
71. min_nb_favourites_per_tweet
72. min_nb_hashtags_per_tweet
73. min_nb_hashtags_per_word_in_the_tweet

74. min_nb_mentions_per_tweet
75. min_nb_mentions_per_word_in_the_tweet
76. min_nb_retweets_per_tweet
77. min_nb_symbols_per_tweet
78. min_nb_symbols_per_word_in_the_tweet
79. min_nb_urls_per_tweet
80. min_nb_urls_per_word_in_the_tweet
81. min_nb_words_per_tweet
82. nb_collected_tweets
83. nb_followees_per_day
84. nb_followers_per_day
85. nb_hashtags
86. nb_hashtags_per_day
87. nb_lists
88. nb_mentions
89. nb_mentions_per_day
90. nb_unique_hashtags
91. nb_unique_mentions
92. nb_unique_sources
93. nb_unique_urls
94. nb_urls
95. nb_urls_per_day
96. numerals_in_screen_name
97. numerals_ratio_in_screen_name
98. portion_of_tweets_with_hashtags
99. portion_of_tweets_with_medias
100. portion_of_tweets_with_mentions
101. portion_of_tweets_with_symbols

102. portion_of_tweets_with_urls
103. proportion_original
104. proportion_replies
105. proportion_retweets
106. replicates
107. replicates_top_20
108. reputation
109. similarity
110. similarity_top_20
111. sources_used_on_average
112. spam_in_screen_name
113. statuses_count
114. std_intertweet_times
115. std_intertweet_times_seconds
116. std_nb_characters_per_tweet
117. std_nb_favourites_per_tweet
118. std_nb_hashtags_per_tweet
119. std_nb_hashtags_per_word_in_the_tweet
120. std_nb_mentions_per_tweet
121. std_nb_mentions_per_word_in_the_tweet
122. std_nb_retweets_per_tweet
123. std_nb_symbols_per_tweet
124. std_nb_symbols_per_word_in_the_tweet
125. std_nb_urls_per_tweet
126. std_nb_urls_per_word_in_the_tweet
127. std_nb_words_per_tweet
128. temporal_bin_0
129. temporal_bin_1

130. temporal_bin_2
131. temporal_bin_3
132. temporal_bin_4
133. temporal_bin_5
134. temporal_bin_6
135. temporal_bin_7
136. time_since_newest_tweet_days
137. time_since_newest_tweet_months
138. time_zone
139. tweeting_frequency_per_day
140. tweets_with_at_top_20
141. tweets_with_hashtags_top_20
142. tweets_with_urls_top_20
143. urls_used_on_average
144. user_id
145. utc_offset

Bibliography

[1] "Facebook: monthly active users 2019 | Statista," 2019. [Online]. Available: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

[2] "Twitter: monthly active users 2019 | Statista," 2019. [Online]. Available: https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

[3] K. Thomas, "The role of the underground economy in social network spam and abuse," Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec 2013. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-201.html

[4] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamalytics: An empirical analysis of spam marketing conversion," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 3–14.

[5] E. Ferrara, "Disinformation and Social Bot Operations in the Run Up to the 2017 French Presidential Election," in *SSRN Electronic Journal*, jun 2017. [Online]. Available: https://www.ssrn.com/abstract=2995809

[6] J. Ratkiewicz, M. D. Conover, M. Meiss, B. Gonc, A. Flammini, F. Menczer, B. Gonçalves, A. Flammini, and F. Menczer, "Detecting and Tracking Political Abuse in Social Media." in *International Conference on Weblogs and Social Media ICWSM*, 2011, pp. 297–304. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewFile/2850/3274

[7] K. Thomas, C. Grier, and V. Paxson, "Adapting Social Spam Infrastructure for Political Censorship," in *5th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2012. [Online]. Available: https://www.usenix.org/conference/leet12/workshop-program/presentation/thomas

[8] N. Abokhodair, D. Yoo, and D. W. McDonald, "Dissecting a social botnet: Growth, content and influence in twitter," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, ser. CSCW '15. New York, NY, USA: ACM, 2015, pp. 839–851. [Online]. Available: http://doi.acm.org/10.1145/2675133.2675208

[9] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016. [Online]. Available: https://m-cacm.acm.org/magazines/2016/7/204021-the-rise-of-social-bots/

[10] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, vol. 6, 2010, p. 12.

[11] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 1–9.

[12] M. McCord and M. Chuah, "Spam detection on twitter using traditional classifiers," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6906 LNCS, pp. 175–186, 2011.

[13] C. Yang, R. C. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers," in *Recent Advances in Intrusion Detection*. Springer, 2011, pp. 318–337.

[14] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," *arXiv preprint arXiv:1701.03017*, 2017.

[15] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '06. New York, NY, USA: ACM, 2006, pp. 16–25. [Online]. Available: http://doi.acm.org/10.1145/1128817.1128824

[16] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "CopyCatch: stopping group attacks by spotting lockstep behavior in social networks," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 119–130.

[17] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering Large Groups of Active Malicious Accounts in Online Social Networks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*. New York, New York, USA: ACM Press, nov 2014, pp. 477–488. [Online]. Available: http://dl.acm.org/citation.cfm?id=2660267.2660269

[18] Y. Li, O. Martinez, X. Chen, Y. Li, and J. E. Hopcroft, "In a World That Counts: Clustering and Detecting Fake Social Engagement at Scale," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016.

[19] K. Darwish, D. Alexandrov, P. Nakov, and Y. Mejova, "Seminar users in the arabic twitter sphere," in *International Conference on Social Informatics*. Springer, 2017, pp. 91–108. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-67217-5_7

[20] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ACM. New York, New York, USA: ACM Press, nov 2010, pp. 35–47. [Online]. Available: http://dl.acm.org/citation.cfm?id=1879141.1879147

[21] F. Benevenuto, T. Rodrigues, J. Almeida, M. Gonçalves, V. Almeida, J. Almeida, and M. Gonçalves, "Detecting spammers and content promoters in online video social networks," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, no. 20080125143100. ACM, 2009, pp. 620–627.

[22] S. Yardi, D. Romero, G. Schoenebeck, and Others, "Detecting spam in a twitter network," *First Monday*, vol. 15, no. 1, 2009.

[23] A. H. Wang, "Don't follow me: Spam detection in Twitter," in *International Conference on Security and Cryptography*, vol. 2010. IEEE, 2010, pp. 1–10.

[24] "Twitter Usage Statistics | Internet Live Stats." [Online]. Available: https://www.internetlivestats.com/twitter-statistics/

[25] "Facebook's Growth Since IPO In 12 Big Numbers | Techcrunch," 2013. [Online]. Available: https://techcrunch.com/2013/05/17/facebook-growth/

[26] "Twitter stock plunges as company blames ad targeting problems for earnings miss," 2019. [Online]. Available: https://www.cnbc.com/2019/10/24/twitter-twtr-earnings-q3-2019.html

[27] "How Does Twitter Make Money?" [Online]. Available: https://www.investopedia.com/ask/answers/120114/how-does-twitter-twtr-make-money.asp

[28] "Twitter Net Worth 2011-2019 | TWTR." [Online]. Available: https://www.macrotrends.net/stocks/charts/TWTR/twitter/net-worth

[29] "Cramer: Twitter's reputation for handling trolls partial to blame for Salesforce loss." [Online]. Available: https://www.cnbc.com/2016/10/17/cramer-twitters-reputation-for-handling-trolls-partial-to-blame-for-salesforce-loss.html

[30] "Twitter earnings Q2 2017." [Online]. Available: https://www.cnbc.com/2017/07/27/twitter-earnings-q2-2017.html

[31] "How Twitter is fighting spam and malicious automation," 2018. [Online]. Available: https://blog.twitter.com/official/en_us/topics/company/2018/how-twitter-is-fighting-spam-and-malicious-automation.html

[32] "Twitter Still Can't Keep Up With Its Flood of Junk Accounts, Study Finds | WIRED," 2019. [Online]. Available: https://www.wired.com/story/twitter-abusive-apps-machine-learning/#

[33] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 591–600.

[34] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *Security and Privacy (SP), 2011 IEEE Symposium on*, IEEE. IEEE, may 2011, pp. 447–462. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5958045

[35] "The New York Times: Fake Twitter Followers Become Multimillion-Dollar Business," 2013. [Online]. Available: http://bits.blogs.nytimes.com/2013/04/05/fake-twitter-followers-becomes-multimillion-dollar-business/?_r=0

[36] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "Towards Detecting Compromised Accounts on Social Networks," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7271060/

[37] ——, "COMPA: Detecting Compromised Accounts on Social Networks." in *Network & Distributed System Security Symposium (NDSS)*, sep 2013.

[38] K. Thomas, F. Li, C. Grier, and V. Paxson, "Consequences of Connectivity: Characterizing Account Hijacking on Twitter," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*. New York, New York, USA: ACM Press, nov 2014, pp. 489–500. [Online]. Available: http://dl.acm.org/citation.cfm?id=2660267.2660282

[39] "Fake Twitter Followers Become Multimillion-Dollar Business - The New York Times," 2013. [Online]. Available: https://bits.blogs.nytimes.com/2013/04/05/fake-twitter-followers-becomes-multimillion-dollar-business/

[40] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on Twitter: human, bot, or cyborg?" in *Proceedings of the 26th annual computer security applications conference*. ACM, 2010, pp. 21–30.

[41] ——, "Detecting automation of twitter accounts: Are you a human, bot, or cyborg?" *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 6, pp. 811–824, 2012.

[42] T. Lokot and N. Diakopoulos, "News Bots: Automating news and information dissemination on Twitter," *Digital Journalism*, 2015. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/21670811.2015.1081822

[43] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna, "Poultry markets: on the underground economy of twitter followers," in *Proceedings of WOSN'12*, 2012, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2342551

[44] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Y. Zhao, "Follow the green: growth and dynamics in twitter follower markets," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 163–176.

[45] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson, "Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse." in *Usenix security*, vol. 13, 2013, pp. 195–210.

[46] "Twitter cuts off API access to follow/unfollow spam dealers," 2019. [Online]. Available: https://techcrunch.com/2019/01/31/dont-buy-twitter-followers/

[47] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi, "Measuring user influence in Twitter: The million follower fallacy," *Icwsm*, vol. 10, no. 10-17, p. 30, 2010.

[48] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi, "Understanding and combating link farming in the twitter social network," in *Proceedings of the 21st international conference on World Wide Web - WWW '12*. New York, New York, USA: ACM Press, apr 2012, p. 61. [Online]. Available: http://dl.acm.org/citation.cfm?id=2187836.2187846

[49] A. Gupta, H. Lamba, and P. Kumaraguru, "$1.00 per RT #BostonMarathon #PrayForBoston: Analyzing fake content on Twitter," in *2013 APWG eCrime Researchers Summit*, 2013, pp. 1–12. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84900812811{&}partnerID=tZOtx3y1

[50] J. M. Berger and J. Morgan, "The isis twitter census: Defining and describing the population of isis supporters on twitter," *The Brookings Project on US Relations with the Islamic World*, vol. 3, no. 20, 2015.

[51] E. Ferrara, "Manipulation and abuse on social media," *SIGWEB Newsletter*, no. Spring, mar 2015. [Online]. Available: http://arxiv.org/abs/1503.03752

[52] X. Zhang and A. A. Ghorbani, "An overview of online fake news: Characterization, detection, and discussion," in *Information Processing & Management*, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306457318306794

[53] A. Bondielli and F. Marcelloni, "A survey on fake news and rumour detection techniques," *Information Sciences*, vol. 497, pp. 38 – 55, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025519304372

[54] P. Meel and D. K. Vishwakarma, "Fake news, rumor, information pollution in social media and web: A contemporary survey of state-of-the-arts, challenges and opportunities," in *Expert Systems with Applications*. Elsevier, 2019.

[55] K. Lee, J. Caverlee, K. Y. Kamath, and Z. Cheng, "Detecting collective attention spam," *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality - WebQuality '12*, p. 48, apr 2012.

[56] "Policy update clarification - research use cases | Twitter Community." [Online]. Available: https://twittercommunity.com/t/policy-update-clarification-research-use-cases/87566

[57] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in Facebook: A case study of unbiased sampling of OSNs," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[58] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@ spam: the underground on 140 characters or less," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 27–37.

[59] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, "Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose," in *International Conference on Weblogs and Social Media ICWSM*. AAAI, 2013, pp. 400–408.

[60] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi, "Rtbust: Exploiting temporal patterns for botnet detection on twitter," in *Proceedings of the 10th ACM Conference on Web Science*, ser. WebSci '19. New York, NY, USA: ACM, 2019, pp. 183–192.

[61] K.-C. Yang, O. Varol, C. A. Davis, E. Ferrara, A. Flammini, and F. Menczer, "Arming the public with artificial intelligence to counter social bots," *Human Behavior and Emerging Technologies*, pp. 48–61, 2019.

[62] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao, "Social Turing Tests: Crowdsourcing Sybil Detection," in *NDSS 2013 (20th Network and Distributed System Security Symposium)*, 2013, pp. 1–14.

[63] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots+ machine learning," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 435–442.

[64] K. Lee, B. D. Eoff, and J. Caverlee, "Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter," in *ICWSM*, 2011, pp. 185–192.

[65] J. Messias, L. Schmidt, R. Oliveira, and F. Benevenuto, "You followed my bot! Transforming robots into influential users in Twitter," *First Monday*, vol. 18, no. 7, 2013.

[66] L. M. Aiello, M. Deplano, R. Schifanella, and G. Ruffo, "People are strange when you're a stranger: Impact and influence of bots on social networks," *Links*, vol. 697, no. 483,151, pp. 1–566, 2012.

[67] C. Cao and J. Caverlee, "Detecting Spam URLs in Social Media via Behavioral Analysis," in *Advances in Information Retrieval.* Springer, 2015, pp. 703–714.

[68] D. Wang, S. B. Navathe, L. Liu, D. Irani, A. Tamersoy, and C. Pu, "Click traffic analysis of short url spam on twitter," in *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on.* IEEE, 2013, pp. 250–259.

[69] S. Lee and J. Kim, "Warningbird: Detecting suspicious urls in twitter stream." in *NDSS*, vol. 12, 2012, pp. 1–13.

[70] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," *Proceedings of the 2011 ACM ...*, pp. 243–258, 2011.

[71] "Confidence in follower counts," 2018. [Online]. Available: https://blog.twitter.com/official/en_us/topics/company/2018/Confidence-in-Follower-Counts.html

[72] "The Twitter celebrities who were hit hardest by the purge of fake accounts," 2018. [Online]. Available: https://qz.com/1327195/the-twitter-fake-follower-purge-hits-company-founders-the-hardest/

[73] N. El-Mawass and S. Alaboodi, "Detecting Arabic Spammers and Content Polluters on Twitter," in *6th International Conference on Digital Information Processing and Communications (ICDIPC'16).* Beirut, Lebanon: IEEE, 2016.

[74] A. Patel, "Marketing "Dirty Tinder" On Twitter," 2020. [Online]. Available: https://blog.f-secure.com/marketing-dirty-tinder-on-twitter/

[75] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Eleventh international AAAI conference on web and social media*, 2017.

[76] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "A Fake Follower Story: improving fake accounts detection on Twitter," *IIT-CNR, Tech. Rep. TR-03*, 2014.

[77] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake twitter followers," *Decision Support Systems*, vol. 80, pp. 56–71, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2015.09.003

[78] A. Alarifi, M. Alsaleh, and A. Al-Salman, "Twitter turing test: Identifying social machines," *Information Sciences*, vol. 372, pp. 332–346, 2016.

[79] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "BotOrNot: A System to Evaluate Social Bots," *arXiv preprint arXiv:1602.00975*, 2016.

[80] S. Y. Bhat and M. Abulaish, "Community-based features for identifying spammers in online social networks," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*, 2013, pp. 100–107. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2492517.2492567

[81] B. Wang, A. Zubiaga, M. Liakata, and R. Procter, "Making the Most of Tweet-Inherent Features for Social Spam Detection on Twitter," in *Proceedings of the the 5th Workshop on Making Sense of Microposts Microposts2015*, 2015.

[82] E. M. Clark, J. R. Williams, C. A. Jones, R. A. Galbraith, C. M. Danforth, and P. S. Dodds, "Sifting Robotic from Organic Text: A Natural Language Approach for Detecting Automation on Twitter," *Journal of Computational Science*, 2015. [Online]. Available: http://arxiv.org/abs/1505.04342

[83] H. Gao, Y. Yang, K. Bu, Y. Chen, D. Downey, K. Lee, and A. Choudhary, "Spam ain't as diverse as it seems: throttling OSN spam with templates underneath," in *Proceedings of the 30th Annual Computer Security Applications Conference*, ACM. New York, New York, USA: ACM Press, dec 2014, pp. 76–85. [Online]. Available: http://dl.acm.org/citation.cfm?id=2664243.2664251

[84] S. Lee and J. Kim, "Warningbird: A near real-time detection system for suspicious urls in twitter stream," *IEEE transactions on dependable and secure computing*, vol. 10, no. 3, pp. 183–195, 2013.

[85] D. Wang and C. Pu, "BEAN: A BEhavior ANalysis Approach of URL Spam Filtering in Twitter," *2015 IEEE International Conference on Information Reuse and Integration*, pp. 403–410, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7301005

[86] A. Aggarwal, A. Rajadesingan, and P. Kumaraguru, "PhishAri: Automatic realtime phishing detection on twitter," in *eCrime Researchers Summit (eCrime), 2012*. IEEE, 2012, pp. 1–12.

[87] A. H. Wang, "Detecting spam bots in online social networking sites: a machine learning approach," in *Data and Applications Security and Privacy XXIV*. Springer, 2010, pp. 335–342.

[88] D. Wang, D. Irani, and C. Pu, "A social-spam detection framework," in *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference on - CEAS '11*. ACM, 2011, pp. 46–54.

[89] M. Jiang, P. Cui, and C. Faloutsos, "Suspicious behavior detection: Current trends and future directions," *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 31–39, Jan 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7389913

[90] I. Inuwa-Dutse, M. Liptrott, and I. Korkontzelos, "Detection of spam-posting accounts on twitter," *Neurocomputing*, vol. 315, pp. 496 – 511, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231218308798

[91] C. Yang, R. Harkreader, and G. Gu, "Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1280–1293, aug 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6553246/

[92] D. Irani, S. Webb, C. Pu, and K. Li, "Study of trend-stuffing on twitter through text classification," in *Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2010.

[93] A. Gupta, H. Lamba, P. Kumaraguru, and A. 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, pp. 729–736. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488033

[94] R. Ghosh, T. Surachawala, and K. Lerman, "Entropy-based classification of'retweeting'activity on twitter," *arXiv preprint arXiv:1106.0346*, 2011.

[95] H. Gao, Y. Chen, K. Lee, D. Palsetia, A. N. Choudhary, and A. Chudhary, "Towards Online Spam Filtering in Social Networks," in *NDSS*, 2012, pp. 1–16. [Online]. Available: http://www.cs.northwestern.edu/{~}ychen/publication-byYear.html{%}5Cnpapers3://publication/uuid/3D6625C7-C9E8-40A8-8A94-872FEF73C0AE

[96] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks," in *2008 IEEE Symposium on Security and Privacy*. IEEE, may 2008, pp. 3–17. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4531141

[97] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 576–589, June 2008.

[98] N. El-Mawass, P. Honeine, and L. Vercouter, "Supervised Classification of Social Spammers using a Similarity-based Markov Random Field Approach," in *Proceedings of the 5th Multidisciplinary International Social Networks Conference - MISNC '18*. Saint-Etienne, France: ACM Press, 2018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3227696.3227712

[99] G. Danezis and P. Mittal, "SybilInfer : Detecting Sybil Nodes using Social Networks," in *Network & Distributed System Security Symposium (NDSS)*, 2009.

[100] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 363–374, 2011.

[101] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering Social Network Sybils in the Wild," 2011. [Online]. Available: http://arxiv.org/abs/1106.5321

[102] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Social fingerprinting: detection of spambot groups through dna-inspired behavioral modeling," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 561–576, 2017.

[103] D. H. P. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining and inference for malware detection," in *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011, pp. 131–142.

[104] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *Proceedings of the 16th international conference on World Wide Web - WWW '07*. Banff, Alberta, Canada: ACM Press, 2007, pp. 201–210. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1242572.1242600

[105] L. Akoglu, R. Chandy, and C. Faloutsos, "Opinion Fraud Detection in Online Reviews by Network Effects." in *proceedings of the international conference on weblogs and social media ICWSM*, vol. 13, 2013, pp. 2–11.

[106] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 985–994.

[107] N. Z. Gong, M. Frank, and P. Mittal, "SybilBelief: A Semi-Supervised Learning Approach for Structure-Based Sybil Detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 6, pp. 976–987, jun 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6787042

[108] P. Gao, N. Z. Gong, S. Kulkarni, K. Thomas, and P. Mittal, "Sybilframe: A defense-in-depth framework for structure-based sybil detection," *arXiv preprint arXiv:1503.02985*, p. 17, 2015. [Online]. Available: http://arxiv.org/abs/1503.02985

[109] D. Hicks and D. Gasca, "A healthier Twitter: Progress and more to do," 2019. [Online]. Available: https://blog.twitter.com/en_us/topics/company/2019/health-update.html

[110] T. Stein, E. Chen, and K. Mangla, "Facebook immune system," in *Proceedings of the 4th workshop on social network systems*, 2011, pp. 1–8.

[111] M. T. Bastos and D. Mercea, "The brexit botnet and user-generated hyperpartisan news," *Social Science Computer Review*, vol. 37, no. 1, pp. 38–54, 2019.

[112] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu, "Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 71–80.

[113] D. M. Freeman, "Can You Spot the Fakes?: On the Limitations of User Feedback in Online Social Networks," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1093–1102.

[114] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "DNA-inspired online behavioral modeling and its application to spambot detection," *IEEE Intelligent Systems*, vol. 31, no. 5, pp. 58–64, 2016.

[115] M. A. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, *Support vector machines*. IEEE, jul 1998, vol. 13, no. 4. [Online]. Available: http://ieeexplore.ieee.org/document/708428/

[116] D. G. Kleinbaum and M. Klein, *Logistic regression: a self-learning text*. Springer Science & Business Media, 2010.

[117] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[118] 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*, vol. 12, pp. 2825–2830, 2011.

[119] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[120] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *International journal of computer vision*, vol. 40, no. 1, pp. 25–47, 2000.

[121] M. Schmidt, "UGM: A Matlab toolbox for probabilistic undirected graphical models." 2007. [Online]. Available: http://www.cs.ubc.ca/{~}schmidtm/Software/UGM.html

[122] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001.

[123] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *Advances in neural information processing systems*, 2005, pp. 1097–1104.

[124] N. El-Mawass and S. Alaboodi, "Data Quality Challenges in Social Spam Research," *ACM Journal on Data and Information Qulaity*, 2017.

[125] A. Weller, K. Tang, T. Jebara, and D. Sontag, "Understanding the bethe approximation: When and how can it go wrong?" in *UAI*, 2014, pp. 868–877.

[126] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1568–1583, 2006.

[127] J. Domke, "Learning graphical model parameters with approximate marginal inference," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 10, pp. 2454–2467, 2013.

[128] D. Koller, N. Friedman, and F. Bach, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[129] V. Stoyanov, A. Ropson, and J. Eisner, "Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 725–733.

[130] S. S. Gross, O. Russakovsky, C. B. Do, and S. Batzoglou, "Training conditional random fields for maximum labelwise accuracy," in *Advances in Neural Information Processing Systems*, 2007, pp. 529–536.

[131] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[132] M. J. Wainwright and M. I. Jordan, "A variational principle for graphical models," in *New Directions in Statistical Signal Processing*. MIT Press, 2005, pp. 21–67.

[133] J. Kim and J. Pearl, "A computational model for causal and diagnostic reasoning in inference systems," in *International Joint Conference on Artificial Intelligence*, 1983, pp. 0–0.

[134] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 736–744, 2001.

[135] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *International Joint Conference on Artificial Intelligence*, 2001.

[136] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, July 2005.

[137] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Map estimation via agreement on trees: message-passing and linear programming," *IEEE transactions on information theory*, vol. 51, no. 11, pp. 3697–3717, 2005.

[138] T. Meltzer, C. Yanover, and Y. Weiss, "Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1. IEEE, 2005, pp. 428–435.

[139] M. Shmidt, "minfunc: unconstrained differentiable multivariate optimization in matlab," http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html, 2005, accessed: 2019-05-12.

[140] N. El-Mawass, P. Honeine, and L. Vercouter, "Champ Aléatoire de Markov pour la Détection Supervisée des Comptes Malicieux sur Twitter," in *20-ème Conférence d'Apprentissage automatique (CAp)*, Rouen, France, Jun. 2018. [Online]. Available: https://hal-normandie-univ.archives-ouvertes.fr/hal-02334326

[141] ——, "SimilCatch: Enhanced social spammers detection on Twitter using Markov Random Fields," *Information Processing & Management*, vol. 57, no. 6, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306457320308128