MACHINE LEARNING-BASED AUTHOR IDENTIFICATION FOR SOCIAL MEDIA FORENSICS

by

Sicong Shao

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2021

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Sicong Shao, titled Machine Learning-based Author Identification for Social Media Forensics and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Salim Hariri — Date: 08/16/2021

Gregory Ditzler — Date: 08/16/2021

Ali Akoglu — Date: 08/16/2021

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Salim Hariri — Date: 8/16/2021
Dissertation Committee Chair
Department of Electrical and Computer Engineering

Acknowledgment

PhD is a long journey that I could not have completed without help from many people.

First of all, my greatest thanks go to my excellent advisor, Dr. Salim Hariri, for his encouragement, guidance, and support during this journey. Thanks to him, I have the opportunity to work in the Autonomic Computing Lab (ACL) and work on meaningful projects. Without him, this work would never have been accomplished.

I sincerely thank Dr. Gregory Ditzler for his help, guidance, and support. He taught me to think across the boundaries of machine learning and cybersecurity. He was always supportive of guided me to realize them concretely.

I would also like to express my thanks to Dr. Ali Akoglu for serving on my qualified, comprehensive, and dissertation exam committees, providing helpful advice and suggestions.

I also would like to thank my minor advisor Dr. Janet Meiling Roveda. I am honored that you were my minor advisor.

I am deeply grateful to Dr. Cihan Tunc for his help, discussions, guidance during my PhD path.

I would like to thank all members of the ACL lab. I also would like to thank Dr. Pratik Satam for his help during my research.

Special thanks to Tami Whelan, Nancy Emptage, Christine Eisenfeld, and Kristi Davenport for administrative help during my PhD study.

Last but not least, I would like to express my gratitude to my parents for their patient, help, and understanding. Without their supports, I would have never gotten here.

Dedications

To my parents.

Table of Contents

List of Figures

List of Tables

Abstract

Social media have gained extreme popularity due to the explosive growth of cyberinfrastructures, mobile devices, Internet technologies, and services. However, they also provide potential anonymity, which in turn harbors hacker forums, carding shops, underground marketplace, dark websites, and so on. As a result, social media have become the playground of cyber threat actors who conduct various malicious operations such as selling stolen cards, disseminating misinformation, propagating hacking tools, spreading malware samples, planning cyberattacks, and organizing trolling campaigns. Therefore, it is urgent to study effective methods that can identify the authors behind the digital text in order to enable forensic analysis, enhance security, and reduce social media misuse. In recent years, machine learning-based author identification has become a promising solution to identify the author of text. However, it is still an underexplored research field in social media forensics. This thesis investigates machine learning-based author identification subfields, including author attribution, author verification, author clustering, and their applications to social media forensics.

Internet Relay Chat (IRC) has traditionally been used for legitimate purposes. Yet, cyber threat actors extensively abuse it to generate a wide range of illegal content and perform malicious behaviors due to its potential anonymity and popularity among hackers. Unfortunately, author identification research in IRC remains a largely underexplored area. In this thesis, we first present our automatic social media monitoring and threat detection method that can effectively collect data for author identification tasks and then present a novel author attribution framework and its application to IRC. It consists of a holistic feature extraction model and an ensemble of ensembles for multi-class classification. We then bring a novel author verification framework under the principle of one-class learning to effectively verify the authorship of IRC texts.

This research also examines author clustering for social media forensics. Most author identification studies focus on author attribution and author verification, while the author clustering research is largely ignored. Meanwhile, cyber threat actors widely make use of

Twitter to create alias accounts for numerous malicious purposes, especially in trolling campaigns and misinformation propagations. Thus, developing an effective author clustering method for Twitter is urgent. In this research, we developed a novel unsupervised learning-based author clustering framework and its application to Twitter. We delivered the capability to identify the group among many Twitter aliases even without prior knowledge of the number of authors.

We address the effectiveness and demonstrate the feasibility of our author identification frameworks through diverse experiments. Our author attribution approach can achieve more than 90% attribution accuracy given hundreds of candidates in the author attribution experiments. In the author verification experiments, over 70% of author cases, our author verification approach can achieve more than 99% AUC. In the author clustering experiments given more than one hundred unlabeled text samples, our author clustering approach attains an average accuracy of 81.93% when knowing the number of authors and an average accuracy of 74.78% without prior knowledge of the number of authors.

1. Introduction

1.1 Overview

As people's lives become increasingly dependent on social media, we are entering a new era in which social media becomes a fundamental part of communication. Despite its numerous irrefutable benefits, cyber threat actors have also exploited social media to create anonymous, fake, and illegal accounts and perform behaviors with malicious intentions, such as misinformation propagation, cyberattacks, cyberbullying, cyberharassment, stolen confidential data trade [1]. Further, these malicious cyber-activists can avoid identification using the methods such as spoofing, VPN, Tor hidden services, and alias accounts [2]. Although the anonymity methods making identification extremely difficult, users still leave some unconsciously identifiable and distinguishable traces, like their writing habits, which can help discover their identities.

As promising techniques to reveal authorships behind texts, machine learning-based author identification methods have gained much attention in recent years [3]. However, most of the current efforts of author identification studies focus on applying machine learning for linguistic research in the digital humanities, such as identifying literary works [4]. In contrast, machine learning-based author identification for social media forensics remains an underexplored area.

IRC is a real-time text-based social media platform that can grant users anonymity by hiding behind aliases that can be changed easily [5]. Although IRC has traditionally been used for legitimate purposes, threat actors have also used it for various malicious purposes due to its potential anonymity [6]. Some IRC channels even offer cybercrime marketing that sells stolen bank accounts, credit cards, hacking tools, zero-day exploits, etc. [7]. Consequently, identifying anonymous users with malicious intentions through author identification in IRC plays a crucial role in understanding, predicting, and preventing cybercriminals' behaviors [5]. An example is that British police caught cybercriminals

actively participating in the organized DDoS campaign “Operation Payback” launched by a hacker group known as *Anonymous* through investigating the conversations in an IRC channel named #AnonOps and then identify them [9].

Another social media platform heavily leveraged by threat actors for conducting various types of malicious behaviors is Twitter [7]. In particular, threat actors may frequently use multiple alias accounts to troll, incorporating fake news, conspiracy theories, and rumors. Further, recent studies show that: A new phase of trolling campaigns emerged when organizations, corporations, and governments sought to influence the opinion regarding important events. As a result, trolling has developed into a severe issue on Twitter. For example, large-scale organized trolls masquerade different users that utilize a wide range of polarized topics to choose from, such as presidential elections, the Black Lives Matter movement, immigration, feminism, and the COVID-19 pandemic [10, 11, 12]. Generally, Twitter reviews suspicious trolling activities to ban trolling accounts and flag/delete misinformation content. These manual solutions have several severe disadvantages, such as subjectivity of judgments, delay in actions, and scaling problems [13].

If the underlying malicious users can be identified, the digital text generated by them on social media could be automatically monitored, warned, alarmed, or blocked. However, because of the anonymity granted by the IRC and Twitter, threat actors can easily create aliases to hide. Further, when they use anonymity techniques to defeat network forensic investigations [3], the digital text left on social media platforms may be the only clues that can be used to identify them.

Machine learning-based author identification can be divided into three aspects:

- Author attribution: Assigning a new digital text of unknown authorship to the most likely candidate when given a set of digital texts of known authorship from a number of candidates [16].
- Author verification: Verifying if a new digital text of unknown authorship is written by that particular author when given only one candidate author who has a set of digital texts of known authorship [14].

- Author clustering: Grouping a set of digital texts of unknown authorship written by their author [15].

These three aspects constitute machine learning-based author identification, enabling author identification to identify text authors in different tasks. This thesis addresses the challenges of building machine learning-based author identification techniques to solve author attribution, author verification, and author clustering for real-world social media forensics tasks. In particular, we present three novel author identification frameworks, including author attribution framework and author verification framework and their applications to IRC, and author clustering framework and its application to Twitter.

1.2 Research Challenge

The practical importance of author identification technologies is obvious for social media forensics. Yet, the general public should also be aware of those technologies' capabilities to make informed and appropriate decisions. This is particularly important because author identification technologies for social media forensics are still underexplored, and active research is needed to push them forward. Author identification for social media forensics have faced a number of challenges:

- **Lack of data for authorship analysis:** Little attention in the literature has been paid to the data collection for authorship analysis. It has resulted in a poor understanding of the properties of writing behaviors. In addition, many previous studies collected data for authorship analysis from asynchronous computer-mediated communication such as forums, websites, newsgroups. By contrast, few studies have addressed the data collection in synchronous mediums for author identification, such as IRC. Further, authorship identification for social media forensics lacks a gold standard benchmark dataset, so comparing performances of different approaches is difficult.
- **Informative feature extraction model:** Authors have unique writing behaviors but share some properties with others from similar backgrounds. It is challenging to define and measure personal characteristics (e.g., the habit of using stop words)

and collective characteristics (e.g., nationality). Further, factors such as personality, interest, knowledge, activity, and cooperation network of authors may have connections to authorship, but little effort has been made to use these possible relationships. Therefore, the optimal writing feature set to capture these relationships has yet to be clarified.

- **Learning attribution model in author attribution:** The machine learning model plays a crucial role in determining the performance of author identification methods. For the author attribution, typically, establishing accurate attribution models requires time-consuming efforts [3]. In particular, the researcher is challenged with selecting suitable models from various machine learning models. Many models, such as Support Vector Machine (SVM), artificial neural networks (ANN), Bayesian models, and Random Forest (RF), are considered as the winners in the different author attribution studies [20, 21]. In practice, the time complexity of models also needed to be considered since the attribution model may deal with large datasets due to a large number of classes (candidates) that appeared in the dataset [22]. In addition, the model should also perform well when the data of each class is limited since it's usually challenging to collect enough data for a single candidate in social media.
- **Learning verification model in author verification**: For the author verification, the challenge is to find the effective model that can identify a particular author's writing amongst all the authors in the given social media forensic data. Many previous works look into the binary classifiers by learning one class of the particular author and the class of all other authors [23, 24]. Yet, it is very challenging or even impossible to collect exhaustive or even representative samples of all other authors. Therefore, it might be better to use a model that can strictly learn the writing of the particular author, and thus, only require the training data from one author. Thus, how to select a suitable model that can learn the normal data distribution strictly is important.
- **Unsupervised learning in author clustering**: For the author clustering, choosing the appropriate clustering model and tuning its hyperparameters for a given author

clustering problem both play crucial roles since different models and hyperparameters often produce drastically different results in clustering [25]. Further, automatically identifying the number of authors (clusters) is considered as one of the most challenging problems of author clustering and even clustering itself [26].

1.3 Contributions

The primary goal of this thesis is to develop machine learning-based author identification techniques consist of author attribution, author verification, and author clustering for different social media forensic tasks. This thesis builds on top of materials from our past papers[14, 15, 16, 17, 18, 19]. The main contributions of the thesis are summarized as follows:

- We developed an autonomic IRC monitoring tool that can collect comprehensive messages for the monitored channel, classify threat levels of messages through recursive deep learning, and intelligently simulate conversation for interacting with users in the channels. This system is important for gathering IRC data for social media forensics.
- We developed a novel feature extraction model to identify authors based on writing behaviors in social media. The feature extraction model incorporated a holistic set of writing features that can be easily adapted to different social media platforms for different author identification tasks.
- We developed the first author attribution version of the deep forest (DF) model, an ensemble of ensembles model that utilizes the fusion of diverse ensemble learning techniques. Our model outperformed state-of-the-art author attribution models, including Multilayer Perceptron (MLP)-based Artificial Neural Network (ANN), Support Vector Machine (SVM), Random Forest (RF), and Multinomial Naive Bayes (MNB), in our experiments.

- We introduced a novel one-class classification-based author verification framework that presents the first text author verification use of the autoencoder-based neural network.
- We developed an automated Twitter user data collection and threat detection tool that can overcome several severe limitations of Twitter API and collect user's rich data (i.e., tweets, metadata, and so on) for threat detection and author identification.
- We introduced a novel author clustering framework whose unsupervised learning unit combines kernel filter, expectation-maximization (EM)-based Gaussian mixture model (GMM), and log-likelihood based cross-validation. Our framework can group authors' writing even without knowing the label and the number of authors.

1.4 Thesis Organization

Following the central themes we have discussed in this chapter, this thesis is organized as follows:

Chapter 2 provides the background material and summarizes the related work related to our research in this thesis.

Chapter 3 presents our automatic social media monitoring and threat detection method on IRC and Twitter.

Chapter 4 presents a novel ensemble learning-based author attribution framework and its application to IRC author attribution.

Chapter 5 presents a new author verification framework under the one-class learning principle and its application to IRC.

Chapter 6 presents a novel unsupervised learning-based author clustering framework for Twitter platforms.

Chapter 7 provides conclusions, contributions, discussion and recommendations for future works.

2. Background and Related Works

In this chapter, we first discuss the background materials about basic concepts that can help understand our research and contributions. We also discuss the related works of author identification and its application to social media forensics.

2.1 Social Media

The term "social media" refers to the means of communications by which users collaborate, share information, and exchange ideas in online communities and networks [28]. The prevalence of personal and sensitive data on social media platforms has made it a breeding ground for cybercrimes and social media misuse. To tackle such occurrences, social media forensics bring justice to victims and combat malicious activities. Social media forensics aims to retrieve social media evidence from user's activities, and evidence of this kind is often crucial to decide whether a person is convicted or acquitted [27].

Using social media data appropriately can give investigators invaluable assistance during the process of the criminal investigation and predict cybercrime since social media provides rich data that may relate to victims and suspects, such as text posts, instant messaging, contact lists, images, videos, location, biography, and so on [27, 124]. The data can also provide specific information about individuals and their connection with others; this information can assist in determining how the cybercrime originated and the malicious purpose and motivation behind it [29].

2.2 IRC

To develop author identification techniques for IRC, it is important to understand the IRC platform and cyber threats and cyber threat actors that occurred in this platform.

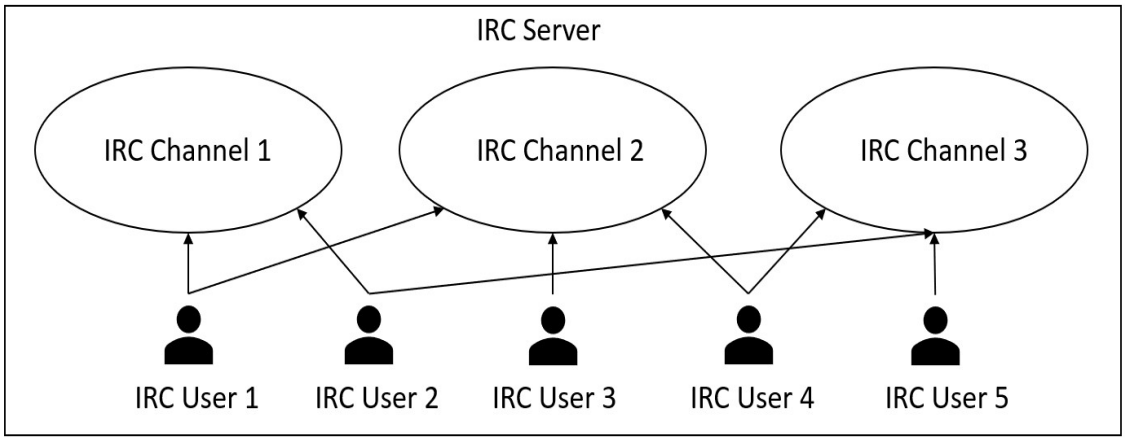


Figure 2.1: The architecture of an IRC server.

2.2.1 IRC Background

Internet Relay Chat (IRC) is one of the earliest social networks for text messaging and can support real-time chat among participants [5]. In spite of the appearance of more sophisticated social networks, IRC remains a popular platform and is still used heavily in many communities, especially the hacker community [6]. There are a large number of IRC servers on the Internet, and a single server may have tens to hundreds of channels with specific topics. Numerous IRC users may congregate in one channel for real-time chats.

The architecture of an IRC server is shown in Figure 2.1. IRC system requires a server that provides networking for connected users through a protocol to facilitate real-time text communications. To provide user anonymity, IRC servers offer the option to users that they can automatically mask their IP addresses when they connect to servers [30]. IRC server provides networks of communication between users in the chat channels publicly as a broadcasted message. IRC users can also identify the receiver for the broadcasted messages, which highlights the messages to the receiver. In the channels, public messages sent by the users are broadcasted to all other users in the same channel in real-time. This differs from the website behavior because on the websites (e.g., blogs), the users can read previously posted messages anytime by browsing them. Hence, contrary to the website

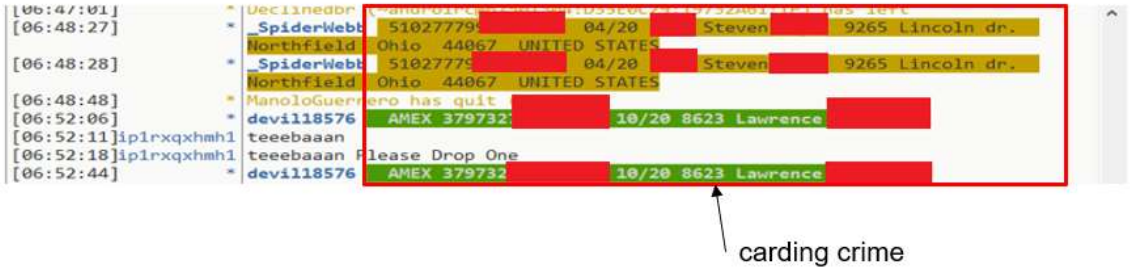
blogs where offline collection and batch processing would work efficiently, the real-time collection is an important research problem in IRC-based communication [5]. In this thesis, to tackle challenges in IRC monitoring, especially for hacker channels, we create an autonomic IRC monitoring tool for the comprehensive real-time recording of the IRC data using several strategies, as discussed in Chapter 3.

2.2.2 Cybersecurity on IRC

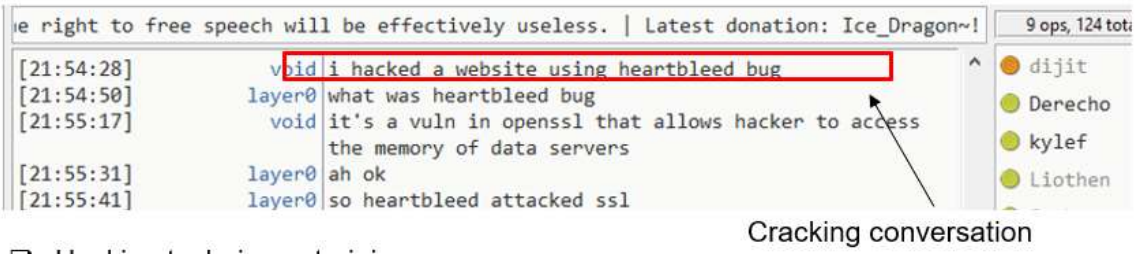
Numerous threat actors congregate within IRC channels due to its popularity among hackers and the anonymity that allows users to create aliases easily [6, 30]. The early use of IRC related to legitimate usages such as discussion, collaborating open-source projects, and teaching technologies; however, some IRC channels eventually became hotbeds of various malicious activities. For example, The #anonops channel is treated as one of the major hacker IRC channels belong to the *Anonymous* hacking organization. In this channel, threat actors frequently broadcast malicious URLs to the dark web. Also, the #anonops channel includes a large number of discussions related to cyber-attacks and hacktivist campaigns. Further, some cybercriminal IRC channels also provide cybercrime markets to sell compromised credit cards (usually compromised from insecure online shopping services), hacking tools, personal information, and stolen credentials. Figure 2.2 shows examples of cyber threats and cybercrime that appeared in IRC channels.

Currently, most social media security research focuses on forum data and ignores IRC data [31]. It may be due to the fact that IRC channels require a connection through specific clients to access the data in real-time; however, data from web-page forums are easier to access without the requirement of collection in real-time. Nevertheless, the importance of IRC is proved by many real-world cases of cybercriminals [5]. More social media forensic research is required for IRC communities.

(a)



(b)



(c)

(d)

Figure 2.2: (a) Example of a cybercriminal IRC channel that sells compromised credit cards, hacking tools, malware, etc. (b) Example of carding crime in an IRC channel. (c) Example of cracking topic discussion in an IRC channel. (d) Example of hacking tool training in an IRC channel.

2.3 Twitter

2.3.1 Twitter Background

The most popular microblogging platform is Twitter, where users can post (i.e., tweets) about almost anything. Tweets are digital texts posted on Twitter to send opinions, status updates, and have conversations. Twitter structures the social connections between users by using followers and followings as account components. The user profile page displays a summary of followers and followers of the user. Tweets from the users and the users they follow are displayed in a customized timeline on their home pages. A tweet can also include hashtags, mentions, replies, and retweets. Hashtags are tags with a form of *#topic* for describing the arbitrary topic. Retweets build on the authority of other Twitter users, denoting the tweet content appeared on other user's timeline. Mentions and replies are represented using *@username*. *Replies* is a special form of *Mentions* with inserting *@username* at the beginning of a Tweet. In addition, Twitter generates a list of topics that are being frequently discussed, called *Trending* topics. This allows users to stay informed about the hot topics of discussion daily. For example, Figure 2.3 shows a participant of an anonymous group. The Twitter account has more than 2,000 followers and promotes hacktivist recruitment and propaganda frequently.

Figure 2.3: An example of a Twitter account operated by a participant of an anonymous group.

2.3.2 Cybersecurity on Twitter

Although there are numerous evident advantages to Twitter usage in social media, Twitter is also heavily used by threat actors. Threat actors have exploited Twitter to perform their varieties of illicit activities, such as spamming, phishing, distributing malware, launching botnet commands, operating (C&C) channels (e.g., IRC-based or HTTP-based), and propagating malicious videos [32]. For example, the Twitter account used by The Shadow Brokers (a hacker group) posted a tweet (shown in Figure 2.4) with a link to the Steem blockchain (a social blockchain that rewards users for sharing content).

Figure 2.4: A tweet from "The Shadow Broker" hacker group claimed they released the link to the leak called *"Lost in Translation"* that includes many exploits, including EternalBlue exploit resulting in WannaCry ransomware that significantly erupted worldwide.

The link to the leak files called *Lost in Translation* includes many exploits developed by National Security Agency. The *EternalBlue* exploits are particularly dangerous among those exploits since they result in the WannaCry ransomware that significantly erupted worldwide.

Moreover, research shows that threat actors tend to form communities that are socially connected. Figure 2.5 present an example of a typical cyber threat actors ecosystem in Twitter. Threat actors extensively spread illegal tweets by adding the link of malicious information sources (e.g., a phishing website) in their tweets. Since victims can automatically receive updates from their following accounts of threat actors, the social connections (i.e., following / follower accounts of each other) and interactions (e.g., retweeting malicious tweets of each other) of threat actors can help them increase the visibility of their malicious tweets and thus gaining more victims.

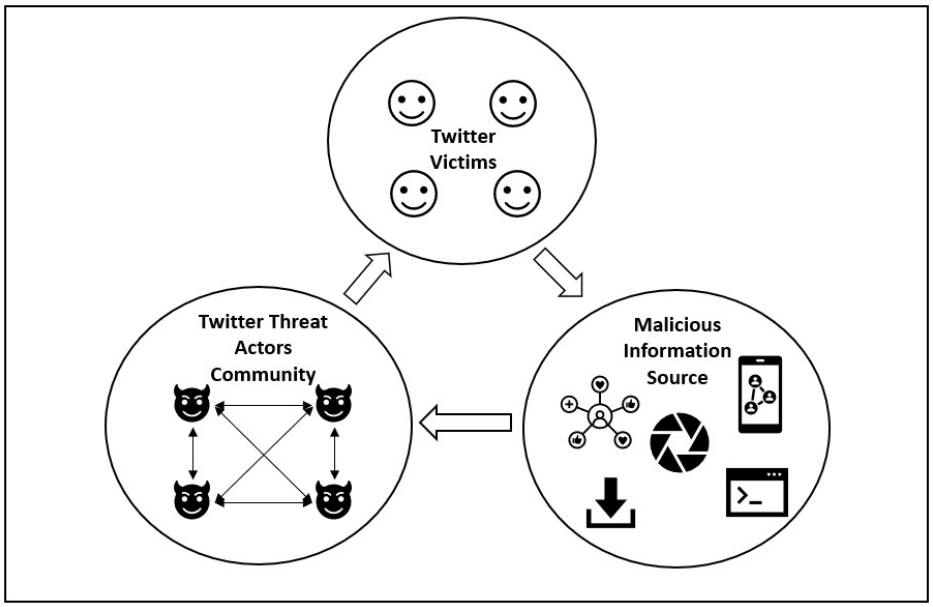


Figure 2.5: A typical ecosystem of cyber threat actor community in Twitter.

In recent years, studies show that organized trolling campaigns turned into a new phase, whereby governments and corporations make efforts to influence the opinion surrounding important events on Twitter [10, 11]. For example, the U.S. Justice Department indicated a Russian state-sponsored media agency named the Internet Research Agency (IRA) to disseminate discord in the US political system. The agency employed hundreds of people to create aliases accounts on social media to promote the Russian government's domestic and foreign policy interests, including Ukraine and the Middle East, and attempt to influence the 2016 US presidential election. Later, Twitter identified 3,814 IRA-linked accounts masqueraded as US citizens to divide voters into a series of issues, such as the Black Lives Matter movement, immigration, and feminism [10].

Recent studies also prove that Twitter has become an important hub for fake news since threat actors distribute a great deal of misinformation on Twitter, intending to deceive users [33]. In addition, automated Twitter accounts (e.g., social bots) also play a critical role in disseminating fake news online by sharing text with misinformation. Recent examples include a large amount of fake news disseminated concerning the prevention, symptoms,

detection, diagnosis, and treatment of the COVID-19 pandemic, which affected society in various ways.

2.4 Related Machine Learning Algorithms

Since author attribution, author verification, and author clustering can be respectively formulated as multi-class classification, one-class classification, and clustering problems. Therefore, for providing concepts of related machine learning algorithms, which are important to understand this research, we introduce the related machine learning algorithms for multi-class classification, one-class classification, and clustering.

2.4.1 Machine Learning Algorithms for Multi-class Classification

This section discusses several important machine learning models, including Support Vector Machine, Random Forest, Multilayer Perceptron-based Artificial Neural Network, and Multinomial Naïve Bayes, which are used as the baseline models in our author attribution tasks.

Support Vector Machine (SVM). There are several hyperplanes that are equally good for evaluating in resubstitution when training samples are linearly separable from the training set. However, these hyperplanes are usually not equivalent in generalization. As we can see from Figure 2.6, an optimal separation is intuitively obtained when the margin to the nearest training samples is as large as possible, because, generally, a larger margin leads to a lower generalization error of a model. Mathematically, SVM [85] models are maximum-margin models. SVM is frequently leverage for the authorship attribution approach. We can define SVM as maximizing the margin between two classes given a dataset of $\left\{(x_i, y_i), i = 1, \dots, n, x_i \in R^d, y_i \in \{+1, -1\}\right\}$, where y_i is the label of class and x_i represent the feature vector. The label of unknown data sample x can be determined by $\hat{y} = sign(w^T \emptyset(x) + b)$ where $\emptyset(x)$is a kernel mapping function and w is the vector that SVM needs to optimize. By calculating the following optimization problem, the optimal hyperplane can be obtained as follows:

$$\begin{gathered} \min \frac{1}{2}(w^T \cdot w) + C\sum_{i=1}^{n} \xi_i \\ s.t.\ \ y_i(w^T \cdot \emptyset(x_i) + b) \geq 1 - \xi_i\ , \\ \xi_i \geq 0, i = 1, \dots, n \end{gathered} \tag{2.1}$$

where ξ is a slack variable, and C is a penalty factor. Its dual form is:

$$\begin{gathered} arg\ \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathrm{K}(x_i, x_j) \\ s.t. \sum_{i=1}^{n} \alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C, i = 1, \dots, n \end{gathered} \tag{2.2}$$

where $\mathrm{K}(x_i, x_j)$ is a kernel function. The classification function is:

$$\hat{y} = sign\left(\sum_{i=1}^{n} \alpha_i y_i K\left(x_i, x\right) + b\right) \tag{2.3}$$

This optimization problem is a quadratic problem that can be solved by a sequential minimal optimization type decomposition method [85].

The binary classification SVM can be extended to multi-class classification by combining some two-category SVM classifiers in a particular manner (e.g., one-against-one), thus forming a multi-class classifier. The one-against-one method for multi-class classification that needs $M(M-1)/2$ classifier for M-class classification [85]. Each classifier is trained on samples from two corresponding classes. A voting mechanism is used for test after all the classifiers are trained. The unknown sample is classified to the class with the largest vote.

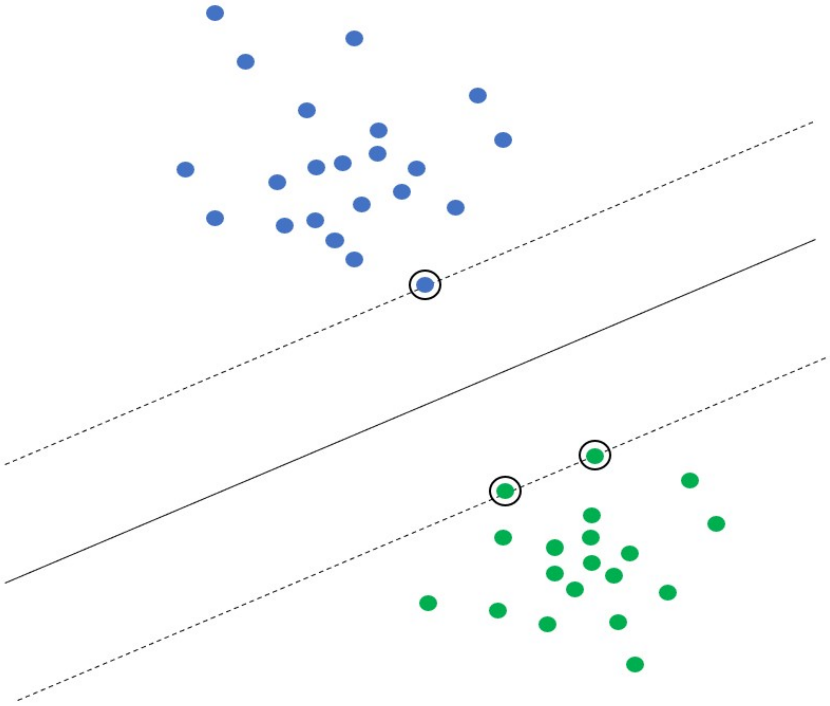


Figure 2.6: An example of an optimal separating hyperplane maximizing the margin to the nearest training samples.

Random Forest (RF). RF [76, 86] consists of t randomized trees. The trees of RF process samples represented by d-dimensional vectors $x \in \mathbb{R}^d$. The training data X^n arriving at the node is partitioned by a splitting function f^n into two subsets $X^n_{f^n=-1}$ and $X^n_{f^n=1}$. A random set of $\mathcal{F}^n$ (splitting functions) for training RF is generated, and the best f^n is chosen based on the impurity measure such as Gini impurity and information gain [76]. The class distribution $P^t_l(m)$ at each leaf node l of a decision tree t is stored. The feature vector of sample is passed by each decision tree t until it reaches leaf node $l(x)$. The class probabilities from all decision trees are averaged. The prediction of classification is given by:

$$m^*(x) = \arg\max_m \frac{1}{T} \sum_t P^t_{l(x)}(m) \,. \tag{2.4}$$

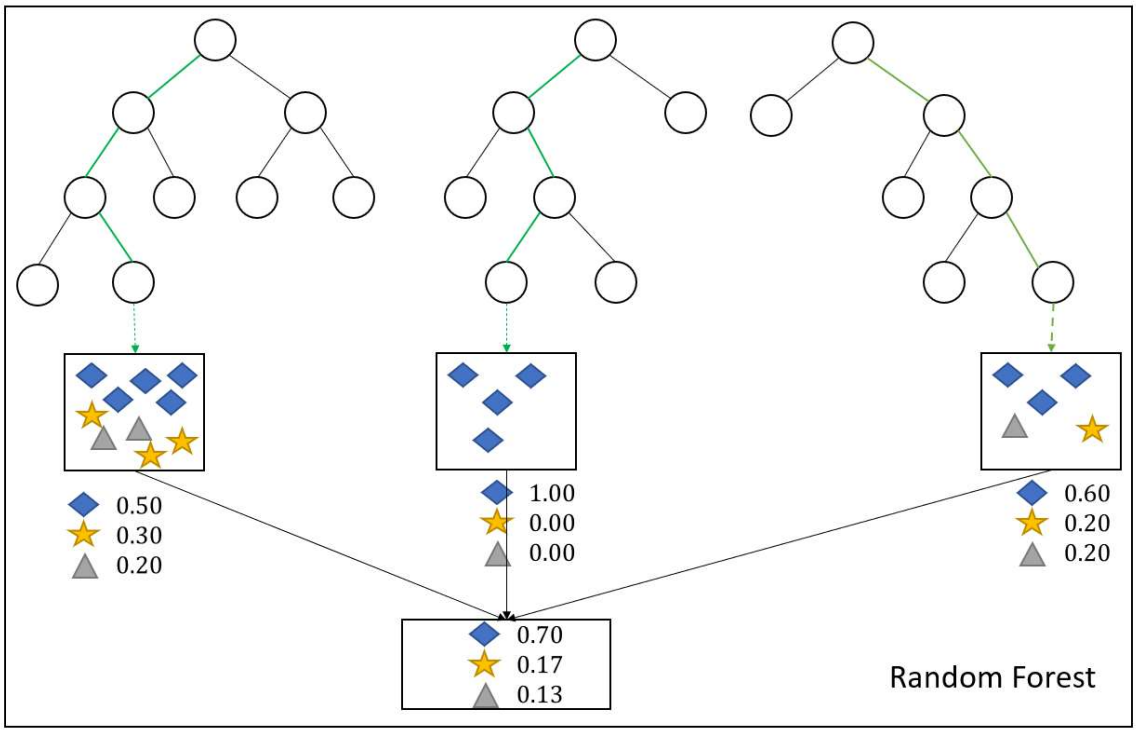


Figure 2.7: An illustration of prediction generation for multi-class classification of Random Forest. The paths of a particular sample traversing through leaf nodes in the trees are highlighted by green color.

As illustrated in Figure 2.7, a tree generates class distribution by calculating the percentage of each class of training samples through the leaf where a particular training sample falls, then RF average the class distribution of all the trees.

Multilayer Perceptron Artificial Neural Network (MLP-based ANN). MLP-based ANN [110] is a common type of feed-forward back-propagation ANN. A basic shallow MLP structure consists of an input layer, an output layer, and a hidden layer. Multiple nodes from each layer are connected to nodes from adjacent layers. Given input variable of a training sample $x = \{x_1, x_2, \cdots, x_d\}$ and a hidden layer having p nodes that are $h = \{h_1, h_2, \cdots, h_p\}$, the weight of a node between x_i and h_j can be represented as w_{ij} and the

bias of the node can be defined as b_j^h. The output of jth node in the hidden layer is formulated as:

$$\varphi_j^h = f\left(\left(\sum_{i=1}^{d} w_{ij}x_j\right) + b_j^h\right) \tag{2.5}$$

where f is a non-linear activation function in the hidden layer. Let w_j^{φ} represent the output weight of jth node between the hidden layer and output layer, and let b^{φ} represent the bias of the output layer. With an activation function g, the output of the basic shallow MLP for binary classification in the output layer is given by:

$$y = g\left(\sum_{j=1}^{p} w_j^{\varphi} f\left(\left(\sum_{i=1}^{d} w_{ij}x_j\right) + b_j^h\right) + b^{\varphi}\right) \tag{2.6}$$

A multi-class classification MLP for classify M classes can be achieved by using M nodes in the output layer. A deeper MLP for multi-class classification can be formed by adding more hidden layers (see Figure 2.8), which enables MLP to learn complicated distribution through in-model feature transformation and layer-by-layer processing.

Multinomial Naïve Bayes (MNB). Naïve Bayes models are learning models with the "naïve" assumption of conditional independence between each pair of features when given the value of the class variable [83]. Multinomial Naïve Bayes (MNB) [84] is one of the Naïve Bayes variants that widely used for text classification. MNB is easy to learn due to its simplicity and efficiency. However, it is obvious that the feature independence assumption is always violated in many real-world applications and therefore affects the performance.

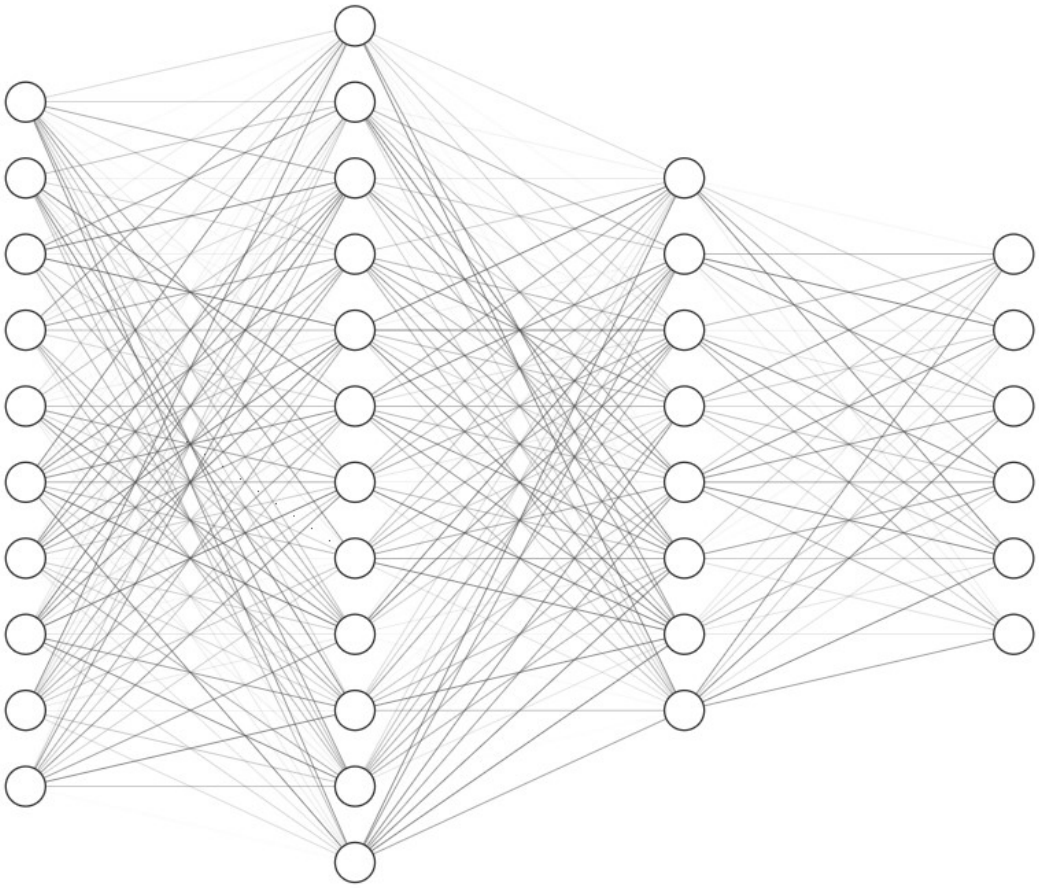


Figure 2.8: An example of a multi-class classification MLP structure consisting of an input layer, an output layer, and two hidden layers.

.

2.4.2 Machine Learning Algorithms for One-class Classification

The author verification tasks can be formulated as either one-class classification problems or two-class classification problems. Here, we focus on the one-class classification problems. Therefore, we focus on introducing several popular machine learning algorithms that perform one-class learning.

Density-based Algorithm. The density-based algorithms assume that samples in low-density regions are regarded as outliers. Local Outlier Factor (LOF) is a typical density-based algorithm [112]. A LOF value is calculated for each sample in LOF. The LOF value

reflects the sparseness of a sample concerning its local neighborhood. Samples having the greatest LOF values are classified as anomalies. Connectivity-based Outlier Factor (COF) improves LOF by examining the connectivity of a sample's neighbors, along with the densities of its neighbors [113]. The anomaly score in COF is computed by the ratio of the mean distance from the sample to its k-distance neighbors and the mean distance from its k-distance neighbors to their k-distance neighbors. Resolution-based Outlier Factor (ROF) defines anomalies as samples that are inconsistent with most of the samples at different resolutions [114].

Isolation-based Algorithm. Isolation Forest (iForest) [101] is a well-known isolation-based algorithm that also can learn one-class data. iForest can build an ensemble of isolation trees, which is used to define the recursive partitioning structure used for isolation. During the iteration of building every isolation tree, iForest randomly selects a feature from the random subset and then selects a random split value between the maximum and minimum value of this feature. The splitting ends up with the condition that every tree node is containing only one sample. The path length is the number of splitting needed to isolate the sample. During the test stage, anomalies are more susceptible to isolation and therefore have a small number of path lengths. Hence, path length can be used as a measure to represent an anomaly.

Kernel-based Algorithm. The most notable example of the kernel-based method is the One-class Support Vector Machine (OC-SVM) [100]. The OCSVM maps input samples into a high-dimensional feature space using a kernel function. The aim is to find the smallest region that includes most one-class samples; the samples that do not fall into this region are considered anomalies.

Reconstruction-based Algorithm. Reconstruction-based algorithms assume that it is more difficult to reconstruct abnormal data than normal data [91]. Some recent work explores the autoencoder for reconstruction-based anomaly detection [130]. The basic idea is to get a compressed representation that better reconstructs the normal sample. Therefore, abnormal samples are more difficult to be reconstructed after projecting them through the compressed representation back to the input representation. Denoising autoencoder, which

is a variant of autoencoder, is also can be used for one-class learning [115]. The denoising autoencoder aims to reconstruct the original input representation by noise perturbation. This mechanism makes denoising autoencoder to extract more robust features of the predictive task reducing the risk of obtaining an identity function. The variational autoencoder extends the standard autoencoder by further finding a distribution that can represent normal samples [116].

2.4.3 Machine Learning Algorithms for Clustering

This section provides the concepts of several important clustering algorithms used in our author clustering problems.

***K*-Means**. *K*-Means is an iterative distance-based clustering algorithm. A prespecified number of clusters are required for *K*-Means clustering. Let $X = \{x_i | i = 1, \ \dots \ , n\}$be a set of samples to be clustered into a set of K-clusters, $C = \{c_k | k = 1, \ \dots \ , k\}$. *K*-Means minimizes the cost function over all *K* clusters as followers:

$$Cost = \sum_{k=1}^{k} \sum_{x_i \in c_k} \|x_i - \mu_k\|^2 \tag{2.7}$$

where μ_k is the mean of cluster c_k. *K*-Means is a greedy algorithm that converges to a local minimum [117].

Hierarchical Agglomerative Clustering. Hierarchical clustering [118] can be either top-down or bottom-up. We use bottom-up in this research. Hierarchical clustering with bottom-up successively agglomerate pairs clusters until all clusters have been merged into a single cluster which includes all samples. Therefore, bottom-up hierarchical clustering is also called hierarchical agglomerative clustering (HAC). When using HAC for flat clustering, we can prespecify the number of clusters *K* and produces *K* clusters.

***X*-Means**. *X*-Means is a regularization framework for estimating k, which extending *K*-Means with efficient estimation of the number of clusters [99]. By extending *K*-Means, it uses statistical criteria to make local decisions maximizing the posterior probabilities of model. *X*-Means searches different values of k and scores each model based on Bayesian Information Criterion (BIC). BIC is a model selection method assuming that one of the models is true, and it tries to find the model which is most likely to be true in the Bayesian view [99].

Gaussian Mixture Model. Gaussian Mixture Model (GMM) [119] is a parametric probability density function represented as a weighted sum of M Gaussian component densities. GMM estimates its parameters from samples using maximum a posteriori estimation or maximum likelihood estimation from a well-trained prior model.

2.5 Personality Insights and Watson Assistant

Since we use Personality Insights [34] to implement the personality analysis for users in IRC and Twitter, and also use Watson Assistant [35] to empower the chat capability of IRC bot. Therefore, in this subsection, we introduce the background of Personality Insights and Watson Assistant.

Watson is the AI platform service provided by IBM to allow users to integrate AI into their applications, training, management, and analysis of data in a secure cloud environment. Watson Assistant is an AI assistant service for social media to answer questions through pre-configured content intents (e.g., banking) [35]. Furthermore, the service can also be improved using interactions history. Another service we leveraged in our approach is the IBM Personality Insights that is based on integrating psychology and data analytics algorithms to analyze the given content and create a personality profile [34]. The IBM Personality Insights service uses three models: Big Five, Needs, and Values. Big Five personality characteristics represent the most widely used model for generally describing how a person engages and interacts with the world. This model includes five primary dimensions based on as follows. (1) *Agreeableness*: a person's tendency to be compassionate and cooperative toward others; (2) *Conscientiousness*: a person's tendency

to act in an organized or thoughtful way; (3) *Extraversion*: a person's tendency to seek stimulation in the company of others; (4) *Emotional range*, also referred to as Neuroticism or Natural reactions: the extent to which a person's emotions are sensitive to the person's environment; and (5) *Openness*: the extent to which a person is open to experiencing a variety of activities. Each of these top-level dimensions has six facets that further characterize an individual according to the dimension. *Needs* model describes which aspects of a product will resonate with a person and includes twelve characteristic needs: Excitement, Harmony, Curiosity, Ideal, Closeness, Self-expression, Liberty, Love, Practicality, Stability, Challenge, and Structure. *Values* model describes motivating factors that influence a person,s decision-making process. The model includes five values: Self-transcendence, Conservation, Hedonism, Self-enhancement, Open to change. Watson infers personality features from textual information using an open-vocabulary approach. Using GloVe, an open-source word embedding technique, the service obtains a vector representation for the words in the input text [34]. It then feeds this representation to a machine learning model that infers a personality profile. To train the model, IBM uses scores from surveys conducted among thousands of users and their Twitter data.

2.6 Author Identification

Machine learning-based author identification attempts to determine the authorship of given texts using machine learning models. Its main assumption is that authors have their unique writing "fingerprint", and therefore, it is possible to identify the authors behind disputed texts by learning their writing style. Machine learning-based author identification generally covers three aspects (i.e., author attribution, author verification, and author clustering), aiming to discover the author of disputed digital texts under different scenarios. With no intention of being exhaustive, this section will review the important related literature works from different author identification subfields.

2.6.1 Author Attribution

Author attribution can be formulated as assigning a new digital text of unknown authorship to the most likely candidate when given a set of digital texts of known authorship from a

number of candidates. The origins of author attribution fall in the area of computational linguistics. However, it can also be applied to forensics, where knowing the authorship of text (e.g., ransom notes) is useful for law enforcement or even saves lives [3]. Segarra et al. used function word adjacency networks to attribute text among ten authors with an accuracy of 93.5% using 100,000 words for each author [4]. Zheng et al developed an author attribution framework based on 270 writing-style features [36]. They attributed up to 20 of the most active users who frequently posted messages in online newsgroups forums, with the best accuracy around 83% when given 20 authors. Abbasi et al. proposed the Writeprints approach based on the extension framework of Zheng et al [2]. Instead of using the same features for all authors, they created individual feature sets for each author according to the individual's key features and executed author attribution experiments with 100 authors on three asynchronous mediums, including email, eBay comments, and Java Forum, with the best accuracy of 85.56%, 94.59%, and 76.87%. Tan and Tsai used a small feature set (50 features) from lexical and syntactic features to identify two selected authors' blog entries with 81.98% accuracy [37]. Solorio et al. extracted writing features from web forum posts from 100 authors. Then, they combined unsupervised and supervised learning to perform author attribution given 5, 10, 20, 50, and 100 authors with an average number of words of 39,664, 40,953, 35,838, 21,502, and 11,322, respectively. Their results achieved accuracy of 76.17%, 77.38%, 71.42%, 63.79%, and 62.10%, with 5, 10, 20, 50, and 100 authors, respectively [38]. Baron et al. presented research on classifiers' evaluation, including naive bayes, decision tree, nearest neighbors, random forest, and multilayer perceptron-based artificial neural network for author attribution [21]. Altakrori et al. benchmark the performance of author attribution approach using a number of popular author attribution classification models with respect to naive bayes, support vector machine, decision tree, random forest, and profile-based model [20].

2.6.2 Author Verification

Author verification can be formulated as verifying if a new digital text of unknown authorship is written by that particular author when given only one candidate author who has a set of digital texts of known authorship. There are several fields in which author

verification techniques is applied. In humanities, author verification can help identify the authors of historical and literary documents that have great impacts. In information security, it allows for the detection of spearphishing attacks and facilitates continuous user authentications. In social media forensics, author verification helps detect aliased accounts. Stamatatos et al. first discussed the author verification problem [39]. According to a dataset of newspaper articles, they applied a regression method to verify authors. Koppel et al. proposed an author verification method called Unmasking, which treats author verification as a one-class classification problem [40, 41]. They use a support vector machine (SVM) classifier to distinguish the text with unknown authorship from a set of known text. Then, they remove the most important feature and repeat this procedure. The unknown text and known text are determined by the same author when the accuracy of SVM significantly drops. Luyckx et al. approximated the author verification as a binary classification problem where they consider available texts by other authors as training samples of abnormal class [23]. Escalante et al. used particle swarm model selection for selecting an ad-hoc classifier for verifying each author in an automatic way [42]. Brocardo et al. extracted stylometry features and then trained a support vector machine model for verifying email and microblog users [24]. Barbon et al. applied the k-nearest neighbors model for performing author verification on Twitter [43]. Litvak used a convolutional neural network model for author verification on an email dataset [44]. Boenninghoff et al. proposed Siamese networks topology for similarity learning on the author verification task [45].

2.6.3 Author Clustering

Author clustering can be formulated as grouping a set of digital texts of unknown authorship written by their author. Author clustering is useful in multiple areas where author information is absent in digital text collections. In a paper submitted anonymously, we might infer that how many people have contributed to writing the paper. In a collection of proclamations written by anonymous groups, it is possible to group proclamations (either from the same group or from different) written by the same author who may participate in multiple groups. In a collection of online anonymous product comments and

reviews, we can conclude that some of the comments and reviews correspond to the same author.

Related works of author clustering are still very limited, especially for the Twitter platform. Luyckx et al. designed an instance of Euclidean distance-based centroid clustering to a collection of literary texts [46]. Iqbal et al. developed an author clustering method for a collection email to extract the writing style feature from suspects and identify the authorship of anonymous messages [47]. Layton et al. presented an ensemble clustering that can estimate the number of different authors through an iterative positive Silhouette approach [48]. Gómez-Adorno et al. performed a hierarchical clustering for clustering authorship of documents, using features including character n-grams, word n-grams, and stylometric features [49].

2.6.4 Author Identification in IRC and Twitter

Even though the importance of author identification techniques is obvious for social media forensics, it is still an underexplored topic for IRC and Twitter. For the author identification in IRC, only author attribution has very few previous works. Layton et al. attributed 50 users in the Ubuntu IRC channel and used inverse author frequency (IAF) weighting and re-centered local profile (RLP) methods to attribute authors [30]. Their results achieved accuracies of over 55% given 50 users. In [50], Inches et al. performed author attribution in the conversation datasets of krijn IRC and IRC-logs separately. The krijn IRC dataset focuses on the topic of the HTML5, and the topics of the IRC-logs dataset range from the programming language and operating system to database and hardware. By applying statistical models with chi-squared distance and Kullback-Leibler divergence, they achieved the best accuracy of 95%, with 343 users in the krijn IRC dataset. However, their accuracy significantly reduced to 61% and 54%, with 148 and 314 users in the IRC-logs dataset, respectively.

There is some research on author attribution and author verification for the author identification in Twitter. In Twitter author attribution, the work of Schwartz et al. proposed a concept called k-signature that uses flexible patterns to attribute authors' texts [51].

Bhargava et al. extract stylometric features and then use RBF to learn the features for Twitter author attribution [52]. Suman et al. presented a Capsule network-based model through character level n-grams feature for Twitter author attribution [53]. In Twitter author verification, Brocardo et al. [24] use the stylometry feature with a feature selection strategy. The study of Alterkavi et al. [54] uses XGBoost to perform feature selection and then uses several classification models such as Logistic Regression, SVM, and Random Forest to construct binary classification-based Twitter author verification models.

Unfortunately, minimal author clustering tasks focus on Twitter. In [55], Yan et al. used character n-grams and function words to extract features of tweets and applied clustering algorithms to identify users. They performed experiments up to 10 users with the best 20.52% accuracy.

3. Automatic Social Media Monitoring and Threat Detection

3.1 Introduction

The development of machine learning-based author identification technologies requires social media data collection since data is crucial to machine learning. The collected digital text information, if analyzed correctly, can offer significant support for author identification tasks since the unique behavior of cyber threat actors can be modeled using data collected from social media. However, collecting and exploring data from cyber threat actors is still an underexplored aspect of social media forensics. In particular, very few studies have addressed data collection efficiency and effectiveness in synchronous computer-mediated communication (e.g., IRC). Due to that, the threat actors can use the latest vulnerabilities to exploit systems (e.g., zero-day attack) and propagate very fast through synchronous communication channels and cannot be normally archived anywhere for later retrieval. Therefore real-time data collection is required for synchronous communication-based social media platforms [5, 18].

Further, many of the current threat detection efforts in social media still depend on keywords search [56]. Keyword search is an effective method to efficiently locate threat information from massive social media data, which is otherwise impossible to locate manually. Yet, the problem is that it may leave out many valuable data or generate too much false information. Moreover, the traditional approaches for threat text detection classify the text into binary classes such as positive and negative. The text, however, can be better scrutinized if they are classified into several threat levels since this will enable investigators to prioritize social media threats. Thus, in order for investigators to properly study cyber threats on social media and understand the motivation of threat actors themselves, how they plan, learn and execute their attacks, developing methods that can provide automatic monitoring, detection, and classification on social media is imperative.

This chapter presents our automatic social media monitoring and threat detection method on IRC and Twitter. We propose autonomic IRC monitoring and automated Twitter data collection that can efficiently and effectively collect the integrity of IRC messages, tweets,

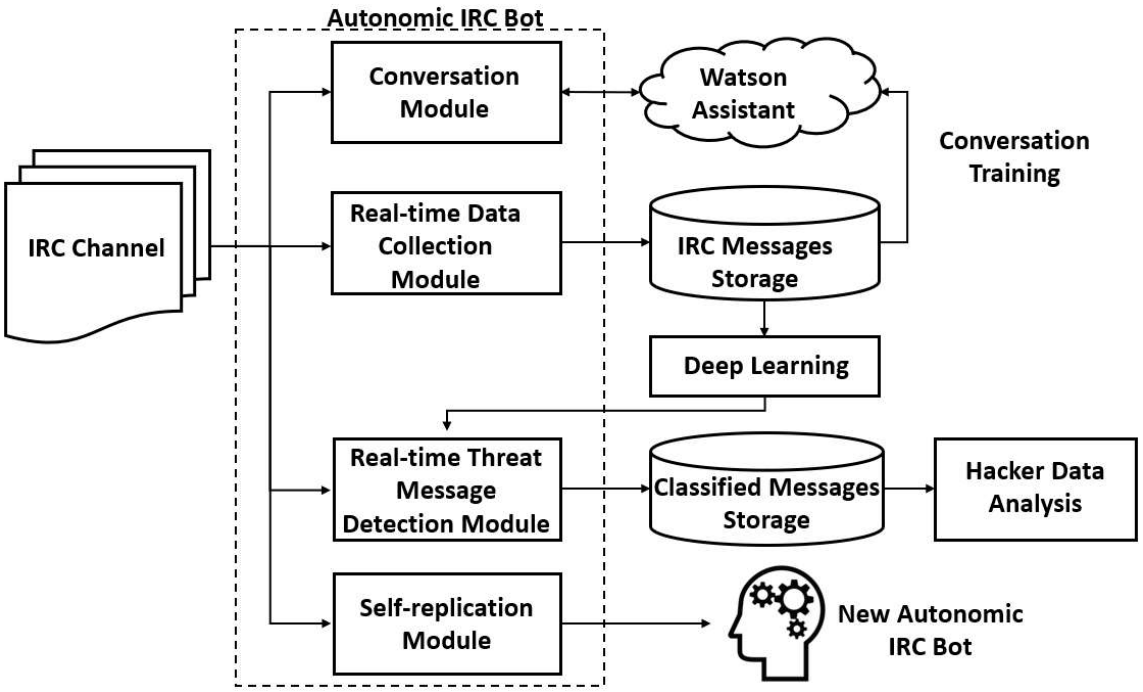


Figure 3.1: Architecture of autonomic IRC bot consists of conversation, real-time data collection, real-time threat data detection, and self-replication modules.

and their metadata. We also apply recursive deep learning to classify digital text into three threat levels that can help investigators by providing intelligent analysis of cyber threats.

3.2 Automated Social Media Monitoring

3.2.1 Autonomic IRC Monitoring

We developed the autonomic IRC bot to collect real-time communication from IRC channels. As shown in Figure 3.1, our autonomic bot consists of conversation, real-time data collection, real-time threat data detection, and self-replication modules. The conversation module is used to provide chat capability through Watson Assistant service [35]. The real-time data collection module is designed to monitor the IRC channel and perform data collection. Furthermore, the real-time threat message detection module can classify the threat level of IRC messages through recursive deep learning (the detail of

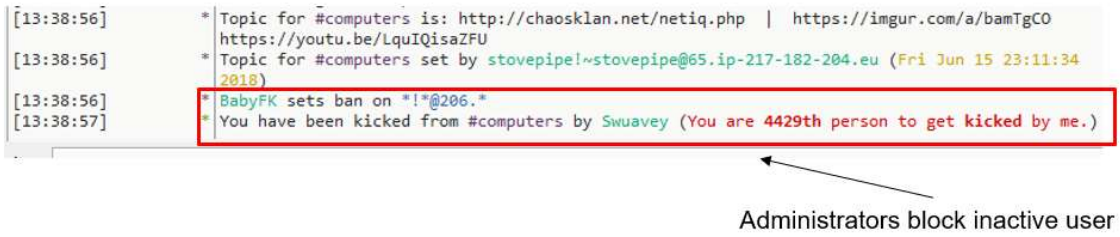


Figure 3.2: An example of an inactive user who was removed and blocked by a channel operator.

recursive deep learning-based threat classification is presented in Section 3.3). The classified IRC messages can be used for hacker data analysis. The self-replication module is designed to create new autonomic bots. For each IRC channel, our bots monitor the IRC data and provide the option to export the collected data to structured data with the following format for analysis: *Threat level* + *User nickname* + *Chat content* + *Date* + *Time.* Our autonomic bot is built with robust continuous monitoring, comprehensive information collection, preprocessing, and classification in real-time.

Concretely, we develop the conversation model to support the real-time data collection module since the IRC channel operators ban users and even block their IP if a user is identified as an inactive user or a bot in some hacker channels. Figure 3.2 shows an example of an inactive user who was removed and blocked by channel operators.

Our IRC bots leverage the intelligent conversation capability that responds to chat like a real user to prevent identification by operators. Using Watson Assistance [35], the bots understand the chat inputs and respond to IRC users by mimicking human conversations as follows: 1) We create a workspace that is a container for dialog flow and training data. 2) The workspace is configured using training data collected from the chat messages. We transform the chat messages into intent, entity, and dialog contents in the workspace for training. The intent is the IRC user's anticipated goal, e.g., Distributed Denial of Service (DDoS). For each intent, we add training samples reflecting what IRC users might ask for the information they need, e.g., "Which tool is available to perform a DDoS attack?". The

Figure 3.3: A hacktivist campaign advertisement that recruits IRC users to join temporary channels for cyberattack.

entity represents a term, an object, or a data type that provides context for an intent. The dialog is a branching conversation flow that defines intents and entities. Through the use of the dialog tool incorporating IRC user's intents and entities, we can add a branch to process every intent that we want the bot to respond to. 3) After the creation of the chat model for the monitored IRC channels, we integrate Watson Assistance into the conversation module of the bot.

Many hacker channels publish self-signed certificates and prevent non-SSL connections. Hence, self-signed certificate creation and port number are added to the real-time data

collection module of our autonomic IRC bot. Also, cybercriminals can create temporary channels to disseminate hacking knowledge or share malicious tools, and even organize cyberattacks. For example, in December 2010, the users in AnonOps IRC server (controlled by a hacking organization known as Anonymous) started using a temporary channel called *#OperationPayback*, which had been quiet for months [57]. In this channel, the cyber attackers discussed their motivation and plan. They then started launching DDoS attacks against the websites of the Swedish Prosecution Authority, everyDnS, Senator Joseph Lieberman, etc., resulting in downtime for these websites. Figure 3.3 shows an example of their recruitment advertisements for the hacktivist campaign called OperationPayback. Hence, to continuously track such activities, a self-replication module is developed to allow the parent bot to generate a new (child) bot that inherits all the parent bot's capabilities.

3.2.2 Automated Twitter Data Collection

Gathering data is also the first step for our author identification and threat detection in Twitter. The traditional way to collect Twitter data is through Twitter API [58]. However, for the studies in Twitter, it suffers several significant drawbacks.

First, its tweet search service can extract tweets by providing keywords with parameters such as language and dates. However, it only returns tweets during the last seven days. This limitation is very challenging for retrospective analyses of investigators for studying social media threats dissemination on Twitter.

Second, the tweet timelines service allows collecting past user tweets. Yet, it can only return up to 3200 of the most recent tweets of an individual account. Thus, it is impossible to retrieve all tweets from a user who posted more than 3 200 tweets. This is a significant drawback for author identification tasks since limited data is provided to model the individual's writing.

Third, the followings and followers service allows retrieving the lists of followers/followings of users. However, by limiting the number of retrieved followers/following per request, Twitter API can only return up to 12,000 different users

from lists of followers/followings in an hour. Assume that if a threat actor has 12,000 followers, this service has to use one hour to collect this threat actor's followers list. The same applies to the user following list. Therefore, it is very time-consuming to use this method to construct the social connection when given a large number of suspects for social network threats analysis. For example, Figure 3.4 shows an influential Twitter account profile from an anonymous group with more than 6.7 million followers. The followings and followers service needs more than 20 days to extract the list of followers, resulting in unacceptable time-consuming.

Therefore, we design an automated Twitter data collection tool to extend the web scraping module Twint [59]. Our data collection tool uses Twitter's HTML pages to collect targeted users' tweets (without limitation of the number of tweets), URL, hashtags, and metadata such as biography, location, and count of likes, replies, and retweets. Also, It can automatically extract the social connection by the followers/followings without the limitation of the number of retrieved followers/following per request. In addition, it can also automatically parses the URL in the tweet content by extracting the title of the pointed source webpage. Further, recursive deep learning is used to perform threat level classification on collected tweet sentences. The storage data can export structured data as a number of different CSV files according to specific tasks such as threat detection and author identification. The overview of the data collection module is shown in Figure 3.5.

Figure 3.4: An influential Twitter account profile from an anonymous group with more than 6.7 million followers.

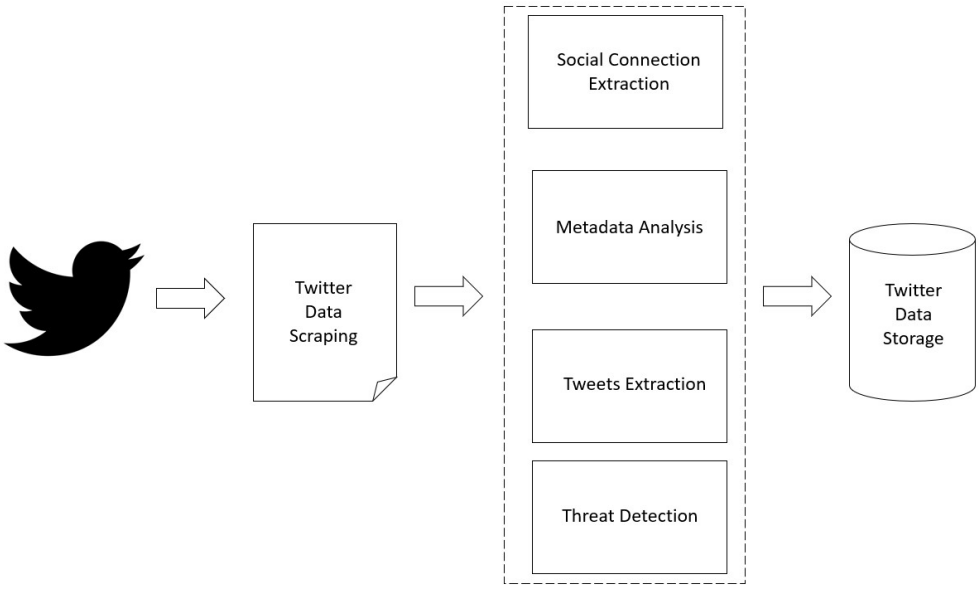


Figure 3.5: Overview of automated Twiter data collection and detection tool.

3.3 Automated Social Media Threat Detection

3.3.1 Recursive Deep Learning for Threat Detection

The recursive neural tensor network (RNTN) has been proven to be powerful for text classification analysis [60]. The RNTN deep learning model represents words and phrases as *D*-dimensional vectors through recursively performing tensor-based composition functions to the parse tree. Words of a sentence are represented as numeric vectors and combined by tensor-based composition functions to form parent vectors in a bottom-up way. The vector of node i is calculated by:

$$V^i = f\left(\begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix}^T T^{[1:D]} \begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix} + W \begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix} + b\right) \tag{3.1}$$

where V_l^i, V_r^i are the vectors of the current node's left child node and right child node. $W \in \mathbb{R}^{D\times 2D}$ is a linear composition matrix, $T^{[1:D]}$ is a tensor, b is the bias vector, and f is a non-linear activation function. The tensor product is given by:

$$\begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix}^T T^{[1:D]} \begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix}^T T^{[1]} \begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix} \\ \vdots \\ \begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix}^T T^{[D]} \begin{bmatrix} V_l^i \\ V_r^i \end{bmatrix} \end{bmatrix} \tag{3.2}$$

where $T^{[d]} \in \mathbb{R}^{2D\times 2D}$ is a slice of tensor $T^{[1:D]}$. The vectors and tensor-based composition function are updated through the backpropagation algorithm [60]. The softmax function is used as the final layer to normalize an input vector into a probability distribution of M class.

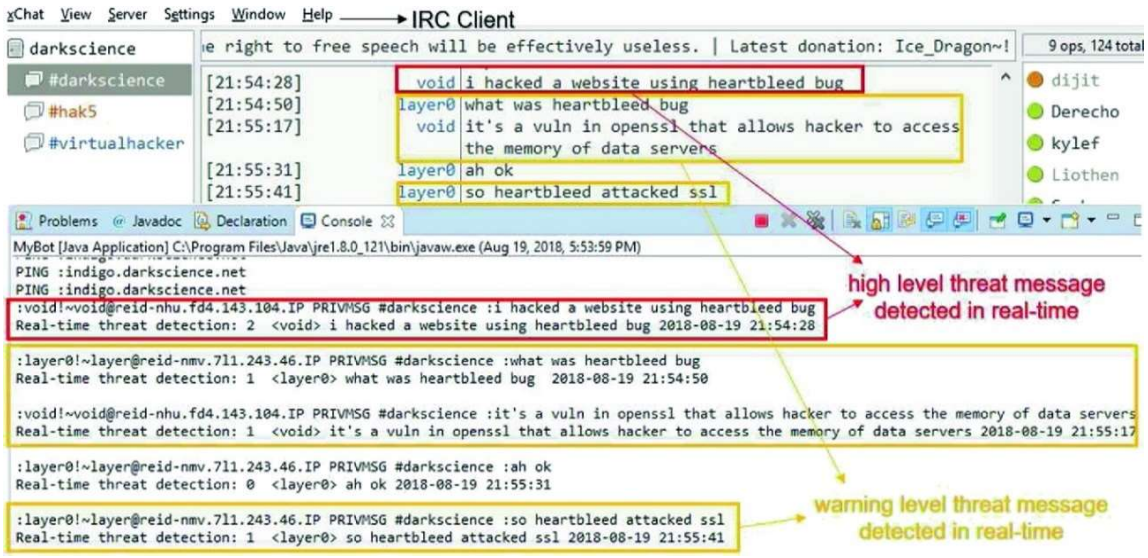
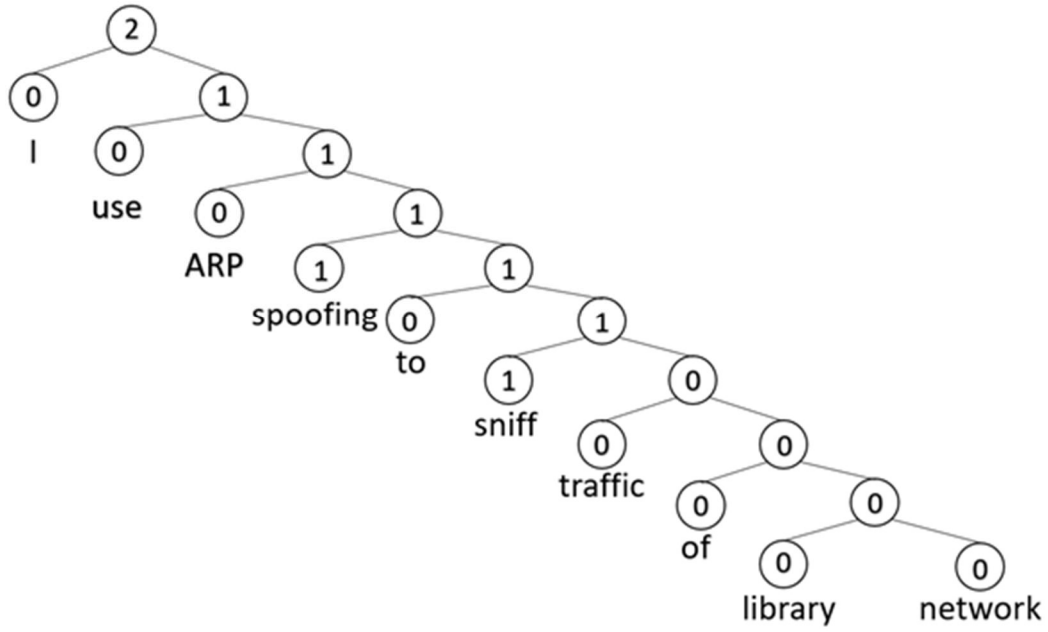


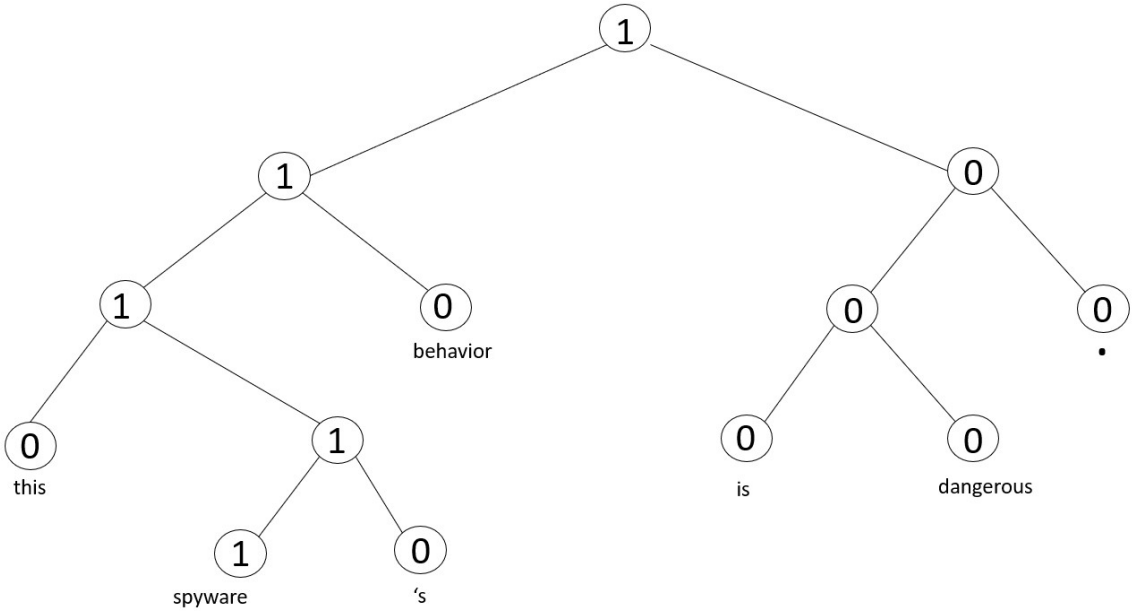
Figure 3.6: An example of real-time IRC message threat level classification in the monitored channel.

Using RNTN, we can automatically distinguish between normal and threat messaging sentence and further identify their threat level. The recursive deep learning model can classify each messaging sentence into three defined levels: Normal (Score 0, for no malicious contents), Warning (Score 1, indicating a potential risk because of the use of hacking terms), and High (Score 2, denoting the sender himself appears to have some malicious behaviors or intentions). Figure 3.6 presents an example of the Heartbleed vulnerability chat topic discussed in the monitored IRC channel. The Heartbleed vulnerability allowed malicious users to read protected memory remotely from many HTTPS websites. As we can see, the autonomic bot classified IRC messages and displayed the threat level of each messaging sentence in real-time.

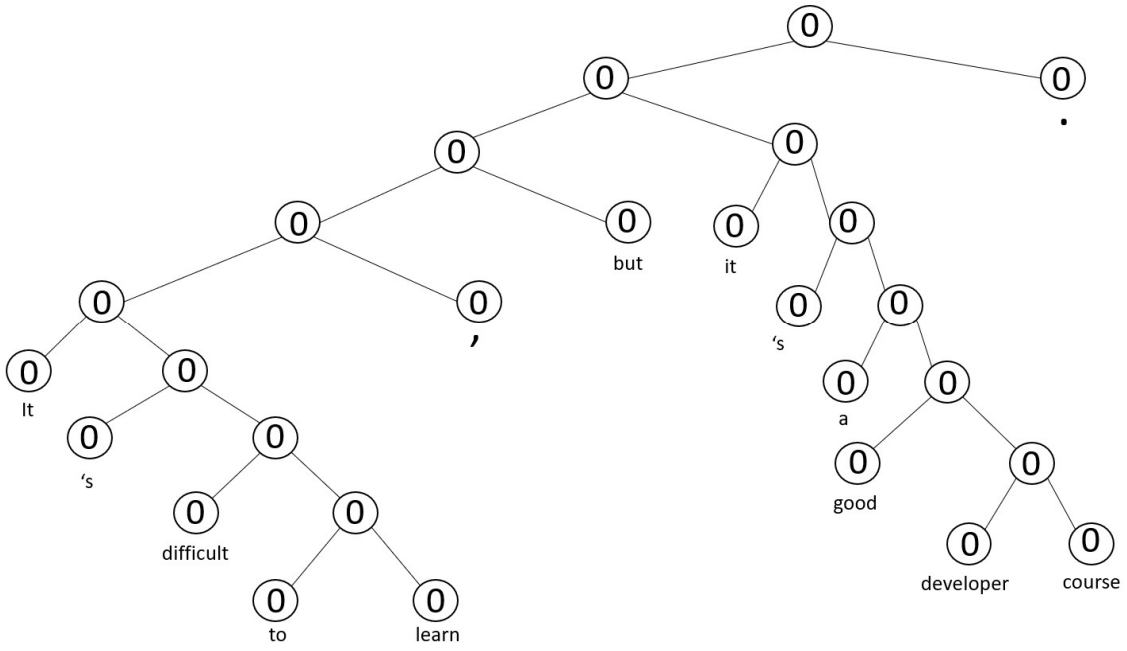
The examples of labeling rules for threat level with Normal, Warning, and High are presented in Figure 3.7. For instance, in Figure 3.7 (a), the representation of “library network” is calculated by the tensor-based composition function of “library” and “network”, and the representation of “of library network” is recursively calculated through the vectors of “of” and “library network”.


2
0
I
1
0
use
1
0
ARP
1
spoofing
1
0
to
1
1
sniff
0
0
traffic
0
0
of
0
0
library
0
network

(a)


1
1
0
1
0
behavior
0
•
0
this
1
0
is
0
dangerous
1
0
spyware
‘s

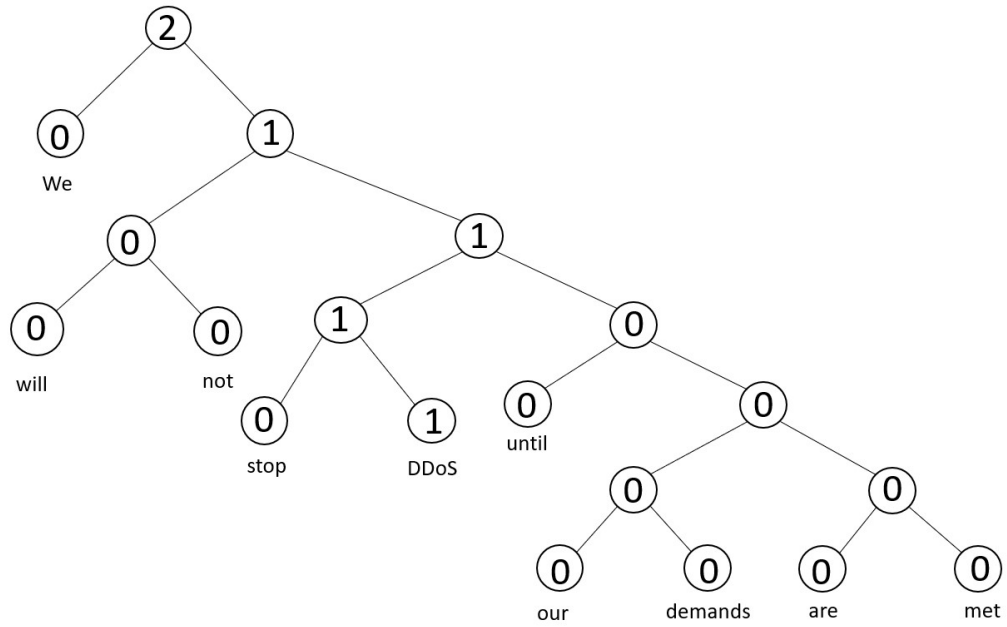
(b)



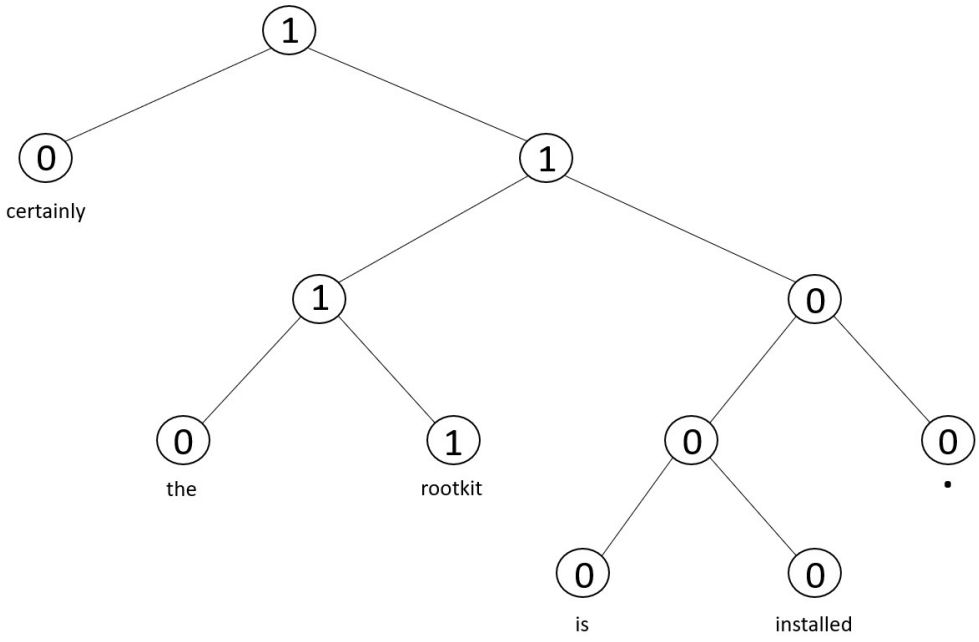
(c)

Figure 3.7: The examples of our labeling rule. Figure 3.7 (a) shows an example of a high threat level messaging sentence, from normal to high (0, 1, 2) at every node of the threat tree construction. Figure 3.7 (b) shows an example of a warning threat level messaging sentence, from normal to warning (0, 1) at every node of the threat tree construction. Figure 3.7 (c) shows an example of a normal messaging sentence, labeling normal at every node of the threat tree construction.

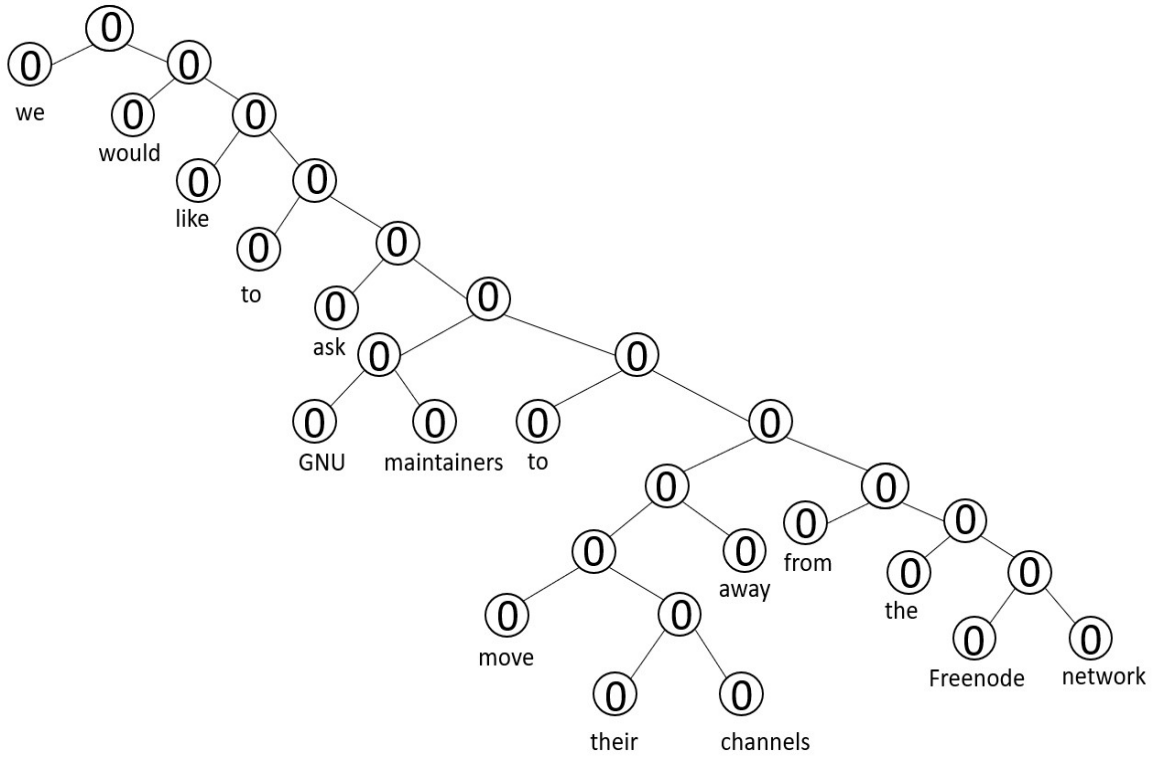
The steps of performing recursive deep learning-based threat classification are listed as follows: (1) Tokenize the message into a sequence of words; (2) Splitting sentence; (3) Tag words and phrases with part of speech tagger (POS); (4) Generates the lemmas (base forms) for words; (5) Parse the sentence into its constituent phrases and words and build a syntactic threat tree; (6) Classify the messaging sentence threat level using the recursive neural tensor network. All the nodes, especially the root node, are given a threat level score. The examples of RNTN threat classification with Normal, Warning, and High are presented in Figure 3.8.


2
0
We
1
0
1
0
0
1
0
will
not
0
1
0
0
stop
DDoS
until
0
0
0
0
0
0
our
demands
are
met

(a)


1
0
certainly
1
1
0
0
1
0
0
0
the
rootkit
.
0
0
is
installed

(b)



(c)

Figure 3.8: The examples of threat classification of RNTN. Figure 3.8 (a) shows an example of prediction for a high threat level messaging sentence. Figure 3.8 (b) shows an example of the prediction of a warning threat level messaging sentence. Figure 3.8 (c) shows an example of the prediction of a normal messaging sentence.

3.3.2 Evaluations

A training set consists of 9,652 IRC messaging sentences was created for training RNTN. A validation set including 1,241 messaging sentences was created for tunning RNTN. A test set containing 906 messaging sentences was created to evaluate the threat level classification of RNTN. The evaluation metric, including Recall and Precision [131] of each threat level in our test set, are shown in Table 3.1. For the Recall metrics, our threat detection model performs well on the normal and warning messaging sentences and produces acceptable results on the high threat messaging sentences. As for the Precision

	Normal (root node)	Warning (root node)	High (root node)
Number in the test set	678	178	50
Percentage in test set	74.83%	19.65%	5.52%
Recall	89.09%	84.83%	76.00%
Precision	98.37%	70.23%	49.35%

Table 3.1: The results of the threat detection model on collected messaging sentences.

metrics, our threat detection model performs excellently on the normal messaging sentences and obtains acceptable performance on the warning messaging sentences. However, for the high-level threat messaging sentences, our model still has much room for improvement. The most important reason for the relatively low precision is the lack of enough training data from the high threat level. Therefore, we believe our threat detection model can be improved after collecting more high-level messages for training. Also, there might be other ways that can lead to better RNTN-based threat detection results; we leave it for future exploration.

3.4 Summary

Current cybersecurity approaches mainly focus on securing computers, networks, and applications from detecting and preventing cyberattacks, mainly in a reactive manner. However, studying the activities of cyber threat actors on social media is also very important to understand their minds and behaviors. Their data stored by data collection tool can help us devise effective author identification mechanisms to identify them to stop these malicious behaviors or mitigate their impacts.

This chapter presents our automatic social media monitoring and threat detection methods in IRC and Twitter. We conduct threat detection experiments and show promising results for threat detection on digital text from social media. Further, our model can also detect

message information based on the malicious intentions of users, which is an unexplored task in previous works.

4. Ensemble Learning-based Author Attribution Framework

4.1 Introduction

There is rapid growth in the use of social networks to disseminate users' opinions and information instantly across the globe [1, 74]. However, the downside of such social networks is that they have also been exploited by cybercriminals to create anonymous, fake, and illegal accounts [61] and to perform illegal activities, including cyberattacks, hacking service trade, drug and weapon marketing, and money laundering [62], such as the Silk Road black market [5]. Furthermore, these illegal cyber-activists can avoid detection using anonymous servers, spoofing, VPN, fake accounts, etc. [3]. Hence, US homeland security and law enforcement agencies have launched multiple projects to prevent deceptive attacks and theft and to track hacker identities [63, 64]. Even though the anonymity methods have made the tracking extremely difficult, users leave some unintentional/unconscious traces, such as their messaging habits, which can be used to identify them. Therefore, when given a number of candidates, author attribution through analyzing attacker messages is one of the most promising solutions [3].

IRC is a real-time communication-based social network platform [5]. Even though IRC has been traditionally utilized for legitimate functions, it has also been the playground of anonymous users for malicious activities. Some channels even contain underground markets selling stolen credit card data, hacking services, security exploits, etc. [31]. Therefore, identifying individual users through author attribution in IRC plays a key role in understanding cybercriminals' behaviors. On the other hand, due to the anonymity provided by the IRC platform, a user can hide behind aliases that can be changed simply. Further, a cybercriminal may leverage anonymity techniques for increasing anonymity to frustrate network forensics. In such a case, the chat messages left on the IRC channels may be the only clue to identify users. Author attribution can discover identities through analysis of users' text. Therefore, author attribution in IRC can play a critical role in determining users.

Performing author attribution in IRC is a challenging task due to the following reasons. First, most of the previous work has focused on asynchronous computer-mediated communication (websites, forums, newsgroups, blogs, comments, etc.) [3]. In contrast, only a few studies have focused on author attribution in IRC, which is a synchronous medium. Second, in the malicious IRC channels, administrators remove/ban inactive users and bots and their IP addresses. This necessitates the development of intelligent and autonomous monitoring systems. Hence, our technique is supported by autonomic IRC monitoring. Third, in most author attribution studies, researchers have focused on author attribution among a limited number of authors up to 100 with significantly reduced accuracy and an increase in the number of authors[2, 3, 20]. For the IRC author attribution task, it is common that the number of candidates is larger with limited training data. Finally, cybercriminals in IRC channels may try to hide their identities to escape cybercrime investigation [65]. As a result, the studies of IRC author attribution are largely underexplored.

This chapter attempts to tackle these challenges by providing a novel ensemble-learning based IRC author attribution framework that can learn the holistic IRC writing feature sets through an ensemble of ensembles model [16].

4.2 Author Attribution Approach

Figure 4.1 shows our IRC author attribution approach architecture. First, using our autonomic IRC bots deployed in various IRC channels, mainly about hacking and cracking activities, we monitor the IRC channels and users' messages and behaviors. The IRC messages of author candidates are extracted to perform two major steps for author attribution: feature extraction and author attribution deep forest (AADF) construction. The AADF construction contains two phases: the feature selection (FS) phase and the attribution model learning phase. After that, the extracted unknown authorship messages represented by selected features can be attributed to the author candidates through the author attribution model.

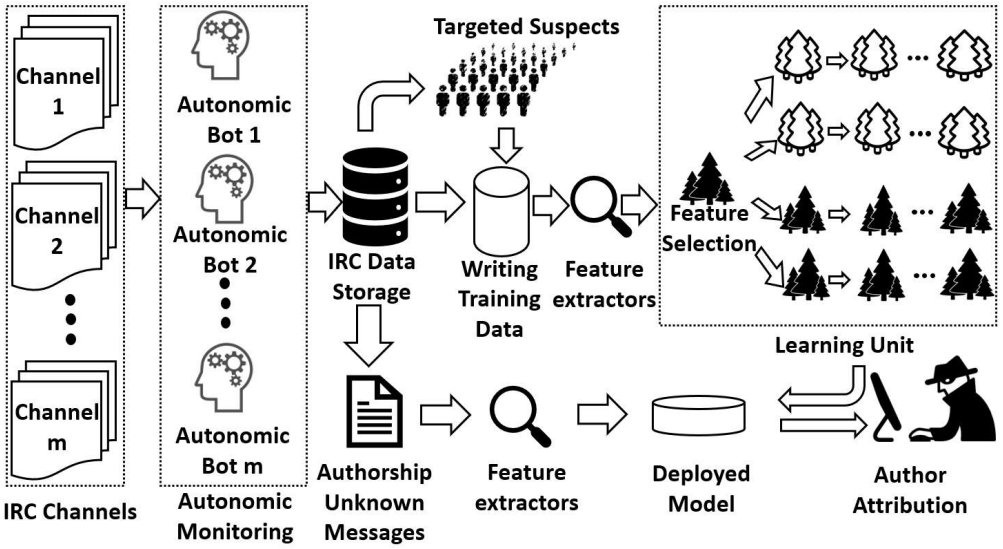


Figure 4.1: Architecture of the IRC author attribution framework.

4.2.1 Feature Extraction Model for Author Attribution

Profile-based author attribution methods and instance-based author attribution methods are two approaches to perform author attribution [66]. The profile-based approach concatenates the available texts of a certain author to one single text file. Therefore, each author has only one big author profile file. The training process of the profile-based approach is simple since the training phase comprises feature extraction from candidate profiles. Then, the author attribution model uses a distance function to compute the most likely author. The majority of modern author attribution approaches consider each text sample as an instance of the known candidate. Unlike the profile-based approach, which has only one long training sample for each suspect, a long text for a certain candidate should be segmented into multiple training samples, probably of equal length in instance-based approaches [66]. In this research, we perform a non-overlapped segment method to each author's whole IRC messages (to create multiple samples), with 1,000 words per sample. Then each sample is used to extract features from the feature set extractors. Table 4.1 summarizes the feature sets used in this study.

Feature Set	Description	The Number of Features
F1: Letters, numbers, special characters, punctuations	Frequency of letters, numbers, special characters, and punctuations	94
F2: Character n-grams	Frequency of the most 500 frequently ranked character 2-grams and the most 1,000 frequently ranked character 3-grams	1,500
F3: Word n-grams	Frequency of occurrence of the most 500 frequently ranked word 2-grams and the most 1,000 frequently ranked word 3-grams	1,500
F4: Stop-words	Frequency of stop-words	543
F5: Cybersecurity knowledge and attitude	Frequency of using NIST key information security terms	6,702
F6: Abbreviation	Frequency of IRC abbreviations	46
F7: Emoticons	Frequency of IRC emoticons	37
F8: Activity Behaviors	Amount of IRC messages in each half-hour	48
F9: Personality Insights	Big Five Model, Needs model, Values Model	52
F10: conversation network	Frequency of sending public one-to-one messages to a candidate	varies

Table 4.1: The Feature Extraction Model for Author Attribution in IRC.

Letters and Numbers and Special Characters and Punctuations. This feature set is one of the most commonly used features in previous research [2, 36]. For each sample, we count the occurrences of 52 letters (case-sensitive), ten numbers, and all the 21 special characters used in the study of Zheng et al. [2]. Since IRC messages might be informal text, the frequency of using punctuation features might also be reliable in IRC author identification. We adopted 11 punctuations (“.”, “,”, “?”, “!”, “‘”, “”, “:”, “;”, “...”, “-”, “—”) in our research.

Character N-grams. The character n-grams feature set is commonly leveraged by authorship analysis tasks [30]. This feature set contains information such as writing behaviors related to capitalization, space, morphological variants of words, typos, and so on. We extracted the most frequent occurrence of character n-grams for feature creation with a frequency ranked cutoff point. Consider the fact that there are cases when the total number of underlying character n-grams types is small due to very limited text samples or frequent repetitive text, which is possible in social media forensics. To make this feature extractor still can work in such cases, we do not consider the low-ranking character n-grams. Therefore, we simply limit the size of this feature set to $n \in \{2, 3\}$ with a frequency ranked cutoff point of 500 for character 2-grams and 1,000 for character 3-grams to make this feature extractor more general. We use more character 3-grams features than character 2-grams due to the fact that the dimensionality of character 3-grams features is much higher than character 2-grams features.

Word N-grams. Users might have the habit of using some phrases constructed by two or more sequential words. Thus, word n-grams features have been effectively used for author attribution [66]. We conjecture that word n-grams would also be helpful for the identification of IRC users. We use the bag-of-words model [66] to represent the feature frequency. Apache Lucene library [67] is used to filter word n-grams. For the same reason as the character n-gram feature set, a ranked cutoff point is set where the 500 most frequently ranked word 2-grams and 1,000 most frequently ranked word 3-grams are selected. This way, each sample contains 500 most frequently ranked of word 2-grams features and 1,000 most frequently ranked of word 3-grams features. Compared to the character n-grams feature set, the word n-grams feature set can capture longer sequential writing information since one word generally contains multiple characters. Another difference is that word-level n-grams are able to extract more semantic meaningful knowledge from IRC messages, while the character-level n-grams can capture uncommon features such as typos and morphological variants.

IRC Stop-words. We explore the stop-words feature set since stop-words typically contain most of the function words (function words determine the syntactical form of a sentence).

Stop-words are generally the most common words in a language; there is no single universal list of stop words used by all NLP tools, and any set of words can be selected as the stop-words for a given purpose [68]. Most search engines generally avoid them due to the high appearance frequency. However, in IRC-based author attribution, we conjecture that stop-words are suitable for IRC authorship analysis since they could provide meaningful information about the IRC users because individuals have their own ways of organizing stop-words. We use the stop-words listed on the website [69], introducing a stop-words list. We measured the frequency of all the IRC stop-words in each user's writing sample to distinguish their characteristics.

Cybersecurity Knowledge and Attitude. The users' cybersecurity knowledge is not equal in IRC channels. Cyber threat actors have different levels of hacking capabilities. Some users are cybersecurity novices and have little cybersecurity knowledge, but others are sophisticated hackers that can utilize advanced cybersecurity techniques. Also, cyber threat actors have different interest topics they like to participate in. While some may be interested in carding, some may be specialized in developing security exploits and recruiting new members. Moreover, the users in the same IRC channel can involve different activities. For example, some may send bot commands to control a channel that authorizes the bot to broadcast certain types of information (e.g., the latest price of bitcoin), some may forward security content links in the channel. Therefore, the information security terms can be used to describe cybersecurity knowledge, the interesting topic, and the activities of monitored individuals. Motivated by helping the user understand information security terminology, NIST has created a repository of key information security terms and definitions in their official webpage of the Computer Security Resource Center [70], which is regularly updated. We extracted 6,702 terms from this glossary and used the occurrence frequency of each term. In order to get the feature value, we use the frequency of occurrence of each term in the NIST glossary and consequently, it enabled us to describe the hacker behaviors using cybersecurity terms.

Abbreviation. Considering the relatively casual environment of IRC, the users' writing styles include abbreviations profoundly. Some users include abbreviations like "afk" (away

afk (away from the keyboard)	asap (as soon as possible)	bbl (be back later)
bbs (be back shortly)	brb (be right back)	bbiab (be back in a bit)
bbiaf (be back in a few)	bbfn (bye bye for now)	btw (by the way)
ctc (care to chat?)	cya (see ya)	FAQ (asked questions)
focl (falling off chair laughing)	fwiw (for what it's worth)	fyi (for your information)
<g> (Grin)	gmta (Great Minds Think Alike)	ic (I see)
imo (in my opinion)	imho (in my humble opinion)	iow (in other words)
irl (in real life)	j/k (just kidding)	ltns (long time no see)
lts (laughing to self)	l8r (later)	lol (laughing out loud)
ly (love you)	mil (mother-in-law)	motd (message of the day)
n/p (no problem)	oic (Oh, I see)	otoh (on the other hand)
ppl (people)	re (RE hi or RE hello)	SAHM (stay at home)
rofl (rolling on floor laughing)	roflmao (rolling on floor laughing)	slm (muslim greeting)
tafn (that's all for now)	ttyl (talk (type) to you later)	wb (welcome back)
ttys (talk (type) to you soon)	wtg (way to go)	wfm (works for me)
WYSIWYG (what you see is what you get)		

Figure 4.2: The used IRC abbreviations for constructing feature extractor F6 *Abbreviation*.

:) (smile)	:-) (smile with a nose)	;) (wink)
;-) (wink with a nose)	:D (very wide grin)	:P (sticking out tongue)
:* (kisses)	:**: (returning kiss)	:I (hmmm...)
: ((frown)	>:((angry frown)	:[(real downer)
:-< (forlorn)	:x (my lips are sealed)	:-o (oh, no!)
:-! (foot in mouth - oops!)	P* (french kiss)	O:) (angel)
[:-| (frankenstein)	^5 (High Five!)	=-O (the enterprise)
:/ (wry)	=:-O (Oh, I'm scared!)	(@_@) (stunned)
(o_o) (shocked)	~\0/~ (waving hello or goodbye)	:-))) (big happy smile)
&:-) (a person with curly hair)	#:-) (from a person with matted hair)	:> (smile with a sarcasm)
:-> (smile with definite sarcasm)	*<|:-) (santa claus or clown)	=^.^= (a cat smiley)
~:-((steaming mad)	_/ (empty glass)	\~/ (full glass)
<:-) (asking dumb question)		

Figure 4.3: The used IRC emoticons for constructing feature extractor F7 *Emoticons*.

from the keyboard), “brb” (be right back), etc., and we have used 46 common abbreviations listed in an IRC beginner website [71], which introduce the commonly used abbreviations (see Figure 4.2) in IRC channels to represent the user’s behavior of typing abbreviations.

Emoticons. The informal nature of IRC creates a need to express a user’s emotions in a textual way; hence, we adopted 37 IRC text emoticon features (see Figure 4.3) from an IRC tutorial website [72]. Thus, the author can be identified by measuring the IRC emoticons feature set.

Activity Behaviors. The activity behaviors of IRC users in the IRC channel are impacted by many factors, such as the living time zone, whether to join regular channel events, and personal activity habits. A user’s activity behaviors can be measured by the number of messages posted in a specific time slot. Therefore, investigating this feature set could unveil clues whether the unknown messages belong to a certain user. Also, variance in the number of messages posted in each time interval can reflect the active degree of users since some users post a little, while other users engrain themselves in the IRC channel and become highly engaged members. Based on this fact, we create time interval features where a day with 48 half-hours is split into 48 equal-size intervals. Then, we count how many messages have been posted in each half-hour interval because we observe that the activity behavior of an IRC user seems to be consistent during half-hour in our monitored channels.

Personality Insights. In the monitored IRC channels, users send messages representing conversation communications. These messages can be measured to characterize user personality. The personality characteristics are distinguished uniquely from individual to individual and relatively stable [73]. Based on how IRC users communicate with others, personality characteristics influence most of the user’s activities and behaviors in the IRC channel. Moreover, personality also influences how IRC users make decisions, including cyberattack types and hacking techniques selection, attack and crime motivation, hacking organizations, and malicious tools being developed, expressed in their messages. Using IBM’s Personality Insights services as explained in Section 2.5, we can successfully analyze individual authors’ IRC messages and intrinsic personality characteristics to create

their personality profiles. Our personality feature extractor can operate using different languages as the Personality Insights service supports multiple languages (e.g., English, Japanese, Korean, Arabic), which is helpful for international cybersecurity investigations. In this thesis, we have focused on author identification with English only, and apply to other languages should be straightforward. By calling the Personality Insights service from IBM Cloud, the personality analysis module can get an individual user's personality insights in JSON format (that has the normalized personality analysis results based on three models: *Big Five*, *Needs*, and *Values*). The *Big Five* model contains the following five primary dimensions: Agreeableness, Conscientiousness, Extraversion, Emotional range, and Openness. Each of these primary dimensions includes six facet features to distinguish a user further. The *Needs* model contains twelve need features, and the *Values* model includes five value features. We selected all the facet features and primary features of *Big Five*, all the features of *Needs* model, and all the features of *Values* model to represent the personality of the suspect user, which creates 52 features in total. A sunburst chart visualization for a user's personality profile is shown in Figure 4.4. Watson recommends that the user provides 1200 words for personality analysis, but providing at least 600 words produces acceptable results.

Conversation Network. In IRC channels, users congregate with conversation networks based on their thread knowledge, capabilities, criminal interests, friendship, etc. While some users are interested in and experienced carding crimes, some specialize in developing vulnerability detection tools and exploitation techniques, and some focus on disseminating hacking tools and recruiting new members. Hence, by studying the pair conversation network of IRC channels, we can build representation of each candidate's pair conversation networks. For public IRC messages, users can point out the receiver's nickname that highlights to whom the messages are being sent. For the candidate sample, we respectively count the number of each candidate's highlighted messages from this certain candidate sample. Therefore, each sample will generate the features which may characterize their unique pair conversation networks in the IRC channel.

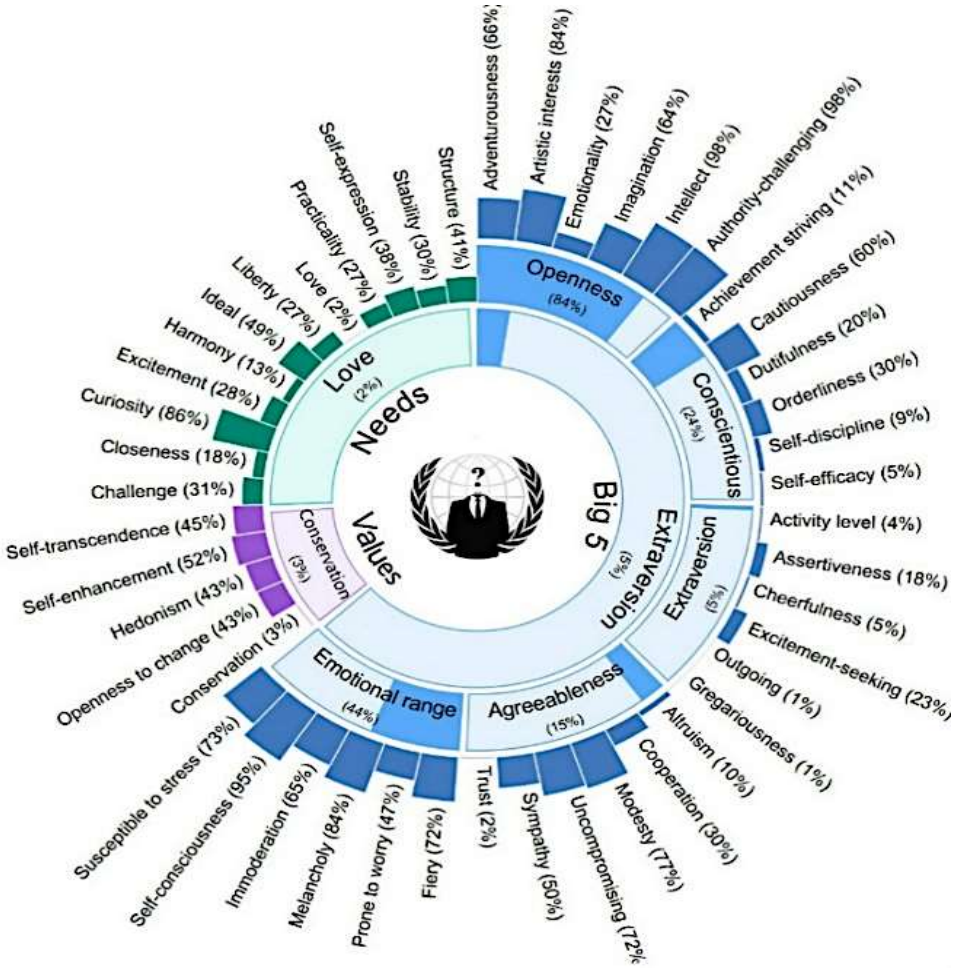


Figure 4.4: Visualization of an IRC user's personality insight features.

4.2.2 Self-adaptive Author Attribution Deep Forest

DF is a tree-based ensemble of ensembles framework that can operate well even with relatively fewer data [75]. By combining ensemble techniques, DF allows layer-by-layer processing, representation learning through the forest, in-model feature transformation, sufficient model complexity, and self-adaptive structure. For IRC author attribution, we propose the first DF-based author attribution approach called author attribution deep forest (AADF). Figure 4.5 shows our self-adaptive AADF architecture. The original version of DF consists of a cascade forest structure (CFS) module and another module called multi-

grained scanning. We replace the multi-grained scanning module with our FS unit because the multi-grained scanning module is used for the analysis of spatial and sequence relationships among raw features. This is unnecessary for our extracted IRC writing style features since they do not convey spatial or sequential relationships.

Feature selection (FS) [125] is an important step in the author attribution machine learning pipeline. The dimensionality of authorship data is usually high (e.g., thousands of features, such as character n-grams, word n-grams, and stop-words, are extracted to represent a single writing sample), whereas irrelevant features may be included resulting in time-consuming in the whole procedure. To reduce this problem, in AADF, the estimates of feature importance can be calculated, and FS can be performed using the tree ensemble model. Thus, feature dimensionality can be significantly reduced while the attribution performance is maintained or even improved.

The most interesting part of DF from the view of author attribution is the CFS, which is a boosting-related and stacking-related procedure. To enhance diversity (highly suggested for ensemble learning), our version uses two types of base tree ensemble models, which are random forest (RF) and extremely randomized trees (ERT).

The RF model consists of an ensemble of decision trees on various sub-samples and uses averaging to improve performance. Also, the best split is determined from a random subset of total features when splitting each node during decision tree construction. The purpose of these two randomnesses is to increase the variance of individual decision trees and, therefore, to decrease the variance of the forest model by combining diverse trees [76].

The ERT model is a tree-based ensemble model for supervised learning [77]. Compared with RF, it is further randomness in the way splits are computed. Instead of searching for the most discriminative thresholds for random subset of candidate features, ERT draws thresholds at random for each candidate feature. Then the best of these random thresholds

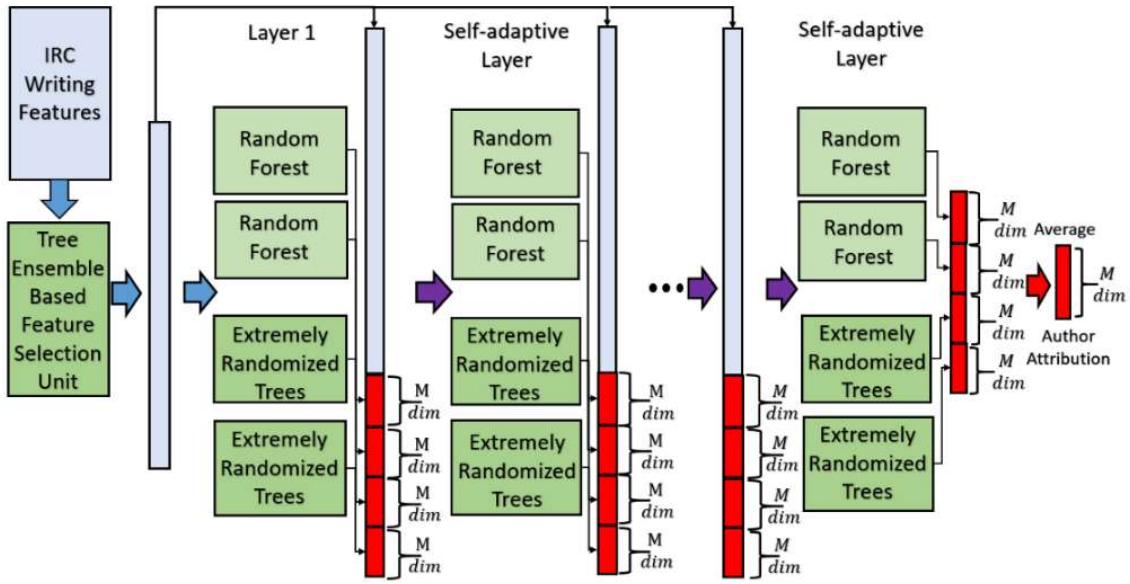


Figure 4.5: Self-adaptive AADF architecture example with base tree ensemble models in each layer and $\boldsymbol{M}$ candidates. Thus, each base tree ensemble model will produce a $\boldsymbol{M}$-dimensional class vector, and each layer will output a $(\boldsymbol{P}+\boldsymbol{Q})\cdot\boldsymbol{M}$ dimensional class vector as augmented features. Here, suppose there are two RF models and two ERT models at each layer of AADF. The augmented features are then concatenated with the output vector of the feature selection unit as the new representation and used as an input vector to each base tree ensemble model of the next layer.

is used as the splitting rule. Therefore, the variance of the ERT model is reduced while its bias is increased.

Let a given training dataset $X = \{x_i, y_i\}(x_i \in \mathbb{R}^d, y_i \in Y,\ |Y| = M, i = 1, 2, 3, \cdots, n)$ be drawn independently and identically distributed (i.i.d) from distribution $\mathcal{D}$, where d is the total number of features after feature extraction, and d' is the total number of features after the processing of FS unit, and M is the total number of classes. Then, the input of CFS is $X_0 \in \mathbb{R}^{n\times d'}$. Each layer has P total RF models and Q total ERT models. Each base tree ensemble model (RF and ERT) can generate a class vector by averaging the class prediction probability across all the trees of that particular base tree ensemble model. Hence, a Z-dimensional class vector is generated as augmented features, where Z is the product of $(P+Q)$ and M. Let l be the index of each layer, where $l = 1, 2, \cdots, h$ and h is the total

number of the layer in the CFS. Except for the layer 1, we call other layers as self-adaptive layers since the number of self-adaptive layers can be automatically determined during the training phase. Except for the final layer, the class vector of the layer l is used as augmented features which are then concatenated with the input X_0 as the new representation and used as input vector to each base tree ensemble model in the next layer. The input data X_l of the layer $l+1$ is given as:

$$V_l = \left\{v_{l,RF_1}, v_{l,RF_2}, \cdots, v_{l,RF_p};\ v_{l,ERT_1}, v_{l,ERT_2}, \cdots, v_{l,ERT_q}\right\} \in \mathbb{R}^{n \times Z} \tag{4.1}$$

$$X_l = \{V_l;\ X_0\} \in \mathbb{R}^{n \times Z'} \tag{4.2}$$

where v_{l,RF_p} and v_{l,ERT_q} are class vectors generated by RF with index p and ERT with index of q in layer l, and $Z' = Z + d'$, and V_l is the augmented feature vector generated by layer l. The class vector generated by each tree ensemble model is produced by k-fold cross-validation. More specifically, each sample will be used as a training sample for k-1 times, generating k-1 class vectors, that are then averaged to generate the class vector, which will be used as augmented features for the next layer. After growing one more self-adaptive layer, the prediction accuracy of the whole cascade structure is estimated by the validation set, and the growth of the cascade structure is automatically terminated if the class prediction accuracy does not increase any further. Hence the total number of layers can be self-determined in a data-dependent way, making the complexity of AADF self-adaptive with the authorship data. Then, the class vector in the final layer h is given as:

$$V_h = \frac{\sum_{\mathrm{p}=1}^{P} v_{h,RF_p} + \sum_{\mathrm{q}=1}^{\mathrm{Q}} v_{h,ERT_q}}{P+Q} \in \mathbb{R}^{\mathrm{n} \times \mathrm{M}} \tag{4.3}$$

Then each sample is attributed to the class with the highest prediction probability in the class vector of the final layer. One of the advantages of the deep-forest-based model compared with most machine learning models is that it is quite robust to the hyper-

parameters setting. In this research, to emphasize that the hyper-parameter setting of AADF is robust and straightforward in author attribution tasks, we use the same setting of CFS across all the experiments of AADF. For simplicity, in AADF, suppose that two RF models and two ERT models are used in the CFS. Each RF contains 500 trees, and each ERT has 650 trees. However, please note that a careful hyperparameter tunning and more computational resources may obtain better author attribution performance, though we have not tried more forests and trees due to the limit of computational resources.

The cascade procedure of AADF is related to boosting since it is able to automatically decide the number of classifiers in the ensemble. Using the output of one level of classifiers as the input to the next level of classifiers is related to stacking. However, traditional stacking is easy to overfit with more than two levels and can hardly enable a deep model [75]. The cascade forest structure aims to solve the problem by concatenating its input X_0 and augmented feature vector V_l generated by layer l as the new input for self-adaptive layer $l + 1$.

4.3 Experiments and Results

4.3.1 Setup

We performed author attribution on hacker IRC channels (shown in Table 4.2). The monitored channels are as follows:

- #anonops: Controlled by the infamous Anonymous hacking organization, which is involved in many cybercrimes [5].
- #2600: A highly active community with hacker magazines and monthly hacker meetings [18].
- #darkscience and #hak5: Hacker communities sharing the interest in security.
- #computer: A discussion channel for the Underworld server for cybersecurity and cybercrimes.
- #thepiratebay.org: Copyright infringement and participating in the swarm for illegal files.

- #trading: An Underworld channel with the topic of black markets.
- #security, #networking, and #bash: Three popular channels involving cybersecurity and computer.
- #bitcoin: Frequent discussing of cryptocurrency usage in black markets and illegal trading.
- #undernet: Hacking and cracking on the Undernet server.

Many studies tend to use over 10,000 words per author for experiments in author identification tasks, which is treated to be a reliable minimum [78]. As the words per text sample, 1,000 words is regarded to be a suitable choice [79]. Hence, we used ten 1,000-word text samples of each user for experiments. As a result, in the monitored 12 IRC channels, the different users who send more than 10,000 words are extracted to experiment with author attribution, resulting in 441 different users to experiment. Each user's whole messages are concatenated and then partitioned to create non-overlapping samples with equal 1,000 words (the remaining messages which are less than 1,000 words are discarded). Next, we apply random undersampling to the corpus [80]. We perform random undersampling to the majority by randomly removing samples from that population until the minority class becomes some specified ratio of the majority class [81]. We set the ratio of the biggest class to the smallest class that is 1:1 to balance the class. Therefore, after random undersampling, each candidate has equal ten 1,000-word samples in the dataset for experiments. Then, the total number of samples is 4,410 with 441 IRC users in the experimental dataset. We named this *dataset 1*.

In order to enhance the diversity of our experiments, the IRC public datasets provided by AZSecure-data project [82] are also used, which include AnonOps, Ed, and Hacker IRC channels. Hacker channel is known for facilitating the activities of the Anonymous hacktivist group. Ed channel has not been originally intended for hacker discussion, but a

Server	Channel	#Messages	Collection Date Range
irc.anonops.com	#anonops	1,326,580	8/15/17 – 9/20/18
irc.2600.net	#2600	785,849	4/01/17 – 9/20/18
irc.darkscience.net	#darkscience	165,896	10/11/17 – 9/20/18
irc.darkscience.net	#hak5	128,389	4/01/17 – 9/20/18
irc.underworld.no	#computer	361,320	9/13/17 – 9/20/18
irc.underworld.no	#thepiratebay.org	81,095	3/13/18 – 9/20/18
irc.underworld.no	#trading	75,996	11/28/17 – 9/20/18
irc.freenode.net	#networking	338,493	9/06/17 – 9/14/18
irc.freenode.net	#security	289,007	4/13/17 – 9/20/18
irc.freenode.net	#bash	266,098	9/06/17 – 9/20/18
irc.freenode.net	#bitcoin	164,663	9/13/17 – 9/20/18
irc.undernet.net	#undernet	310,723	12/05/17 – 9/20/18

Table 4.2: Total Number of Collected Messages in the Monitored IRC Cybersecurity Channels.

large number of hackers and hacktivists use it to communicate and share knowledge, due to its popularity and anonymity [82]. All the channels' data are collected from September 2016 to May 2018. For the AnonOps channel, the data after August. 15th. 2017 is dropped to avoid overlap with our collected AnonOps channel data in *dataset 1*. For the Hacker and Ed channels, all the data is considered. Then we use the same strategy used for creating *dataset 1* to create *dataset 2*, resulting in 185 different users with ten 1,000-word samples for each user. The descriptions of *dataset 1* and *dataset 2* are presented in Table 4.3.

Dataset	Authors	Features	Samples per Authors	Words per Samples	Samples
Dataset 1	441	10,963	10	1,000	4,410
Dataset 2	185	10,707	10	1,000	1,850

Table 4.3: Description of Experimental IRC author attribution Datasets.

4.3.2 Results

In this section, we validate the effectiveness of AADF approach and compare AADF with competing approaches. We first investigate the attribution performance of different approaches without FS unit. After that, we compare AADF to other approaches with tree ensemble-based FS units.

We use all the feature set extractors described in Table 4.1 to perform feature extraction. The min-max normalization [83] is applied for feature scaling. We compare the 10-fold cross-validation results of state-of-the-art author attribution classification models to the AADF approach. The support vector machine (SVM), naive bayes (NB), and random forest (RF) are considered as state-of-the-art author attribution classification models [20]. Artificial neural network (ANN) is also used for comparison due to that it is also high success in the author attribution domain [21] as well as on various tasks. For SVM, we use the linear kernel, radial basis function kernel, polynomial kernel, and sigmoid kernel, respectively. For NB, we use the multinomial NB (MNB) classifier because it is suitable for text classification [84]. The DF's base ensemble model called RF is also an effective model for author attribution. We use multilayer perceptron-based ANNs having one layer, two layers, and three layers, respectively. All the ANNs use fully-connected layer and ReLU as the activation function and Adam as the optimizer. For our experiments, we used a Dell PowerEdge R710 (with 2 Intel Xeon X5660 CPUs and 128GB RAM) running Ubuntu 18.10 operating system. The implementations of MNB, RF, and ANN are based on Scikit-Learn [83]; the implementation of SVM is based on LIBSVM [85];

The implementation of AADF is based on Scikit-Learn and gcForest package [75]. For the AADF approaches, 3-fold cross-validation is used for producing class vector. The number

of self-adaptive layer is automatically determined. The training set is split into two parts, including growing set and estimating set. We call the subsets generated from the training set as growing/estimating set to avoid confusion with the validation sets of 10-fold cross-validation. Then we use the growing set to grow the cascade, and the estimating set to estimate the prediction accuracy. The self-adaptive layer growing process is automatically terminated when the prediction accuracy on the estimating set is not improved. Then, the cascade is retrained based on the training set which is merged by growing set and estimating. Accuracy is used to measure the results of different approaches. We calculate the average accuracy for each approach using the 10-fold cross-validation method with 4,410 samples for *dataset 1* and 1,850 samples for *dataset 2*.

To study the attribution performance of the different approaches on the original dimensionality of datasets, we run the experiments on *dataset 1* and *dataset 2* using the approaches without FS unit. Table 4.4 shows the results of different approaches without using FS unit. As we can see from the results, AADF clearly outperformed other approaches on the original dimensionality. After we investigate the *dataset 2*, we found that the important reason why its accuracy is lower than *dataset 1* is that many different active authors in *Ed* channel heavily perform similar or even same post-flood that involves posting large numbers of posts with repetitive text in the channel, which increases the author attribution difficulty.

After studying the attribution performance of the approaches with original dimensionality, we evaluate the approaches with FS unit. We used ERT and RF and gradient boosting machine (GBM) to automatically select features. All of these can measure the importance score of each feature to learn the impact of features towards predicting the classes. In order to encourage diversity, which is crucial to ensemble construction [86], instead of using 500 trees and 650 trees for RF and ERT in the CFS layer, the tree ensemble models in the FS unit use 700 trees for RF and ERT. The importance value of a feature is calculated by mean decrease impurity (MDI) for FS using RF and ERT. A GBM framework called LightGBM

Dataset 1					Dataset 2				
MNB	RF	ANN	SVM	AADF	MNB	RF	ANN	SVM	AADF
69.14	93.45	92.95	93.38	**95.24**	63.68	90.70	91.35	90.27	**92.11**

Table 4.4: Comparisons of Accuracies of Tenfold Cross-Validation of Different Approaches without FS

developed by Microsoft [87] is used to achieve GBM. Gradient boosting decision trees (GBDT) and gradient-based one-side sampling (GOSS) are respectively used as the boosting type of the GBM to perform FS. GBDT is a tree-based ensemble learning method that is trained in sequence by fitting the residual errors. GOSS is an enhanced GBDT method using sampling that randomly ignores training samples with small gradients. Both of GBDT and GOSS FS units use 100 tree estimators. The author attribution performance has not improved further after adding more tree estimators for FS (and more trees also require more training time and increases the computational time). The GBM achieved by LightGBM can measure feature importance through the parameter called split, which is the number of times the feature used in the GBM. For the RF and ERT-based FS, the feature importance values of all features are normalized and summed to 1. As the GBM FS methods, LightGBM keeps the number of times the feature used as the feature importance values (split) and does not normalize them. If a feature's importance value is smaller than the defined threshold, then this feature is discarded and not considered as important. A number of different thresholds of feature importance values are used to test the performance of different approaches. We set the minimum value of the threshold as 1 for GBM methods, and 10^{-7} for RF and ERT methods, to extract all the features with non-zero feature importance value when setting the thresholds as the minimum. For each round of 10-fold cross-validation, FS measures the feature importance using training data of that round. Next, the training data with selected features (by FS) is used to train the learning model, and then, the model is evaluated using the selected features on the validation set of

that round. The average accuracy of 10-fold cross-validation is used to measure the performance of different approaches.

To study the performance of AADF with FS unit, we run experiments using AADF with FS unit and also compare AADF to different approaches with FS unit. Tables 4.5– 4.8 present the accuracy of each approach with varying methods of FS by applying different feature importance thresholds. Table 4.9 and Table 4.10 show the average number of selected features of each FS with different feature importance thresholds. We observed that the ERT-FS improved accuracy of AADF on *dataset 1*, and all the FS methods in this study are able to improve AADF accuracy on *dataset 2*. This shows that our FS units are useful for AADF. Further, by using ERT-FS with the same MDI threshold 10^{-6} , AADF attains the best accuracy results with 95.42% on *dataset* 1 and 92.70% on *dataset 2*. The average numbers of selected features are 5011.9 on *dataset* 1 and 4667.8 on *dataset 2*, that enables AADF to achieve the best accuracy results through ERT-FS. Hence, AADF can use ERT-FS to significantly reduce the feature dimensionality, while its attribution performance is even improved. Among non-AADF approaches, the ANN achieves the best accuracy results with 93.90% on *dataset 1* and 92.00% on *dataset 2*. All the FS methods in this study improve the accuracy results of the MNB, SVM, and ANN approaches on both datasets. The RF-FS is able to improve the accuracy of RF on *dataset 2*. As we can see from the results, for all the FS units with all the feature importance thresholds used in the experiments of this study, AADF achieves better accuracy than other approaches. Hence, we conclude that AADF outperforms other approaches with FS units in this study.

MDI	Dataset 1					Dataset 2				
	MNB	RF	ANN	SVM	AADF	MNB	RF	ANN	SVM	AADF
10^{-3}	65.15	90.63	85.76	79.50	**91.72**	63.35	85.62	81.30	75.30	**87.35**
10^{-4}	86.42	92.40	93.74	93.47	**94.72**	83.89	89.68	91.84	90.32	**92.16**
10^{-5}	85.90	92.99	93.74	93.67	**94.76**	80.54	90.32	90.70	90.22	**92.27**
10^{-6}	84.97	92.88	93.17	93.38	**95.08**	78.59	90.76	91.03	90.32	**92.65**
10^{-7}	84.97	92.99	93.06	93.33	**94.81**	78.59	90.70	90.54	90.32	**92.54**

Table 4.5: Comparisons of Accuracies of Tenfold Cross-Validation of Different Approaches with RF-FS

MDI	Dataset 1					Dataset 2				
	MNB	RF	ANN	SVM	AADF	MNB	RF	ANN	SVM	AADF
10^{-3}	43.42	88.34	82.74	67.21	**89.37**	41.89	78.38	74.86	55.24	**79.62**
10^{-4}	87.73	92.88	93.56	93.61	**94.29**	84.97	90.32	91.62	90.49	**92.27**
10^{-5}	85.71	92.99	92.88	93.40	**94.69**	80.49	90.49	91.30	90.32	**92.11**
10^{-6}	84.10	93.13	93.29	93.31	**95.42**	78.38	90.38	90.92	90.27	**92.70**
10^{-7}	84.06	93.27	93.36	93.36	**95.22**	78.32	90.59	90.86	90.27	**92.54**

Table 4.6: Comparisons of Accuracies of Tenfold Cross-Validation of Different Approaches with ERT-FS

Split	Dataset 1					Dataset 2				
	MNB	RF	ANN	SVM	AADF	MNB	RF	ANN	SVM	AADF
100	74.01	89.43	86.60	82.88	**90.45**	83.14	89.46	91.14	89.14	**92.11**
50	81.54	91.34	90.73	89.41	**93.11**	84.70	90.00	91.89	89.84	**92.22**
10	87.39	92.31	93.47	93.40	**94.58**	83.95	89.84	90.81	90.16	**92.05**
5	87.60	92.77	93.54	93.40	**94.69**	84.11	90.27	91.30	90.16	**92.38**
1	86.83	92.86	93.76	93.42	**94.76**	83.51	90.38	91.68	90.43	**92.54**

Table 4.7: Comparisons of Accuracies of Tenfold Cross-Validation of Different Approaches with GBDT-FS

Split	Dataset 1					Dataset 2				
	MNB	RF	ANN	SVM	AADF	MNB	RF	ANN	SVM	AADF
100	70.95	89.52	85.28	81.54	**89.89**	83.03	88.65	91.03	88.49	**92.00**
50	79.41	90.93	89.98	88.68	**93.02**	84.86	89.89	91.03	89.57	**92.22**
10	86.96	92.59	93.36	93.42	**94.51**	84.81	90.11	92.00	90.43	**92.27**
5	87.53	92.54	93.90	93.49	**94.54**	84.16	89.84	91.19	90.11	**92.43**
1	86.67	92.88	93.67	93.58	**94.40**	83.41	90.43	91.51	90.32	**92.32**

Table 4.8: Comparisons of Accuracies of Tenfold Cross-Validation of Different Approaches with GOSS-FS.

MDI	Dataset 1		Dataset 2	
	RF-FS	ERT-FS	RF-FS	ERT-FS
10^{-3}	166.6	86.5	141.3	49.1
10^{-4}	2,291.8	2,738.1	2,323.7	2,750.5
10^{-5}	4,054.1	4,485	4,039.2	4,159.2
10^{-6}	4,731	5,011.9	4,528.6	4,667.8
10^{-7}	4,736.7	5,022.2	4,528.8	4,668.7

Table 4.9: Average Number of Selected Features of Tenfold Cross-Validation of Different Feature Importance Threshold with RF-FS and ERT-FS

Split	Dataset 1		Dataset 2	
	GBDT-FS	GOSS-FS	GBDT-FS	GOSS-FS
100	292.5	249.7	1,317.3	1,270.8
50	727	639.2	1,944.5	1,889.8
10	2,208.9	2,111.9	2,612.6	2,646.3
5	2,676.6	2,628.4	2,815.9	2,932.2
1	3,649.3	3,621.3	3,190.5	3,226.6

Table 4.10: Average Number of Selected Features of Tenfold Cross-Validation of Different Feature Importance Threshold with GBDT-FS and GOSS-FS.

To understand the contribution of different feature sets, we investigate the selected number of features and the percentage of usage of each feature set. Table 4.11 presents the average number of selected features and the total number of features of that feature set when performing 10-fold cross-validation experiments. For *dataset 1* and *dataset 2*, the cases of RF-FS and ERT-FS with a threshold of 10^{-6}**,** GBDT-FS with a threshold of 1, and GOSS-FS with a threshold of 5 are selected to present because AADF respectively achieved the best accuracy with them in RF-FS, ERT-FS, GBDT-FS, and GOSS-FS. As we can see, all the feature sets contribute features to the attribution model with the best performance in each FS unit. In addition, the feature sets of character n-grams (F2), word n-grams (F3), IRC stop-words (F4), and IRC cybersecurity knowledge and attitude (F5) contribute more features in each type FS method, since a lot of these features are included in their feature sets before FS. From the perspective of efficiency, we notice that IRC personality insights feature set (F9) achieved the best efficiency because all of them are selected by the four different FS methods for both datasets. Figure 4.6 shows the statistic results of the F9 of 4,410 samples on *dataset 1* before feature rescaling (min-max normalization), that can help researcher understand hacker behaviors through personality insights. The IRC activity behaviors (F8) and letters, numbers, special characters, punctuations (F1), are also highly efficient feature sets. They have over 95% usage by using different FS methods. The feature set of IRC public pair conversation network (F10) has good usage for RF-FS and ERT-FS on both datasets. For *dataset 1*, F10 has over 68% representation with ERT-FS and RF FS, whereas it attains over 50% usage by GBDT-FS and over 30% usage by GOSS-FS. For the *dataset 2*, F10 feature set has over 77% representation by using ERT-FS and RF-FS, and it gets over 40% usage by GBDT-FS and over 30% usage by GOSS-FS. The explanation of the good usage of this feature set is that IRC users form their pair conversation networks based on friendship, common topics of interest, specialization, and location, which can help to characterize the users. We also observed that the IRC abbreviation feature set (F6) and the IRC emoticons feature set (F7) have good usages for RF-FS and ERT-FS while low usages for GBDT-FS and GOSS-FS. F6 gets over 49% usage using RF-FS and ERT-FS methods, and over 18% usage by GBDT-FS and GOSS-FS methods for both datasets. F7 approach over 40% representation by RF-FS and ERT-

Feature Set	Dataset 1					Dataset 2				
	Total	RF	ERT	GBDT	GOSS	Total	RF	ERT	GBDT	GOSS
F1	94	92	92	92	91.9	94	92	92	91.1	90.2
F2	1,500	1,500	1,500	1,496.6	1,480.1	1,500	1,498.5	1,499	1,490.4	1,487.6
F3	1,500	1,397.2	1,487.3	840.3	307.3	1,500	1,353.9	1,396	768.9	640.6
F4	543	484.2	505.5	418.2	275.7	543	492.3	498.9	391.3	364.9
F5	6,702	815.5	953.5	454	212.7	6,702	809	894	249.6	172.7
F6	46	22.7	28.1	14.4	8.6	46	24	24.9	11.6	10.4
F7	37	16.2	17	13.9	10	37	15.8	15.3	7.5	5.9
F8	48	48	48	46	46	48	48	48	46	46
F9	52	52	52	52	52	52	52	52	52	52
F10	441	303.2	328.5	221.9	144.1	185	143.1	147.7	82.1	61.9

Table 4.11: Average Number of Selected Features Using RF-FS and ERT-FS with Threshold $\mathbf{10^{-6}}$ and GBDT-FS with Threshold 1 and GOSS-FS with Threshold 5.

FS, and at least 15% representation using GBDT-FS and GOSS-FS for both datasets. Due to the large number of n-grams features used by AADF, it is natural to ask which n-grams features most frequently appear. Thus, the top 10 most frequent n-grams of each n-grams feature set in our study are listed in Tables 4.12 – 4.13. It is interesting to note that the word 3-grams statistic reflects how frequently the similar or same flood posts appeared in *dataset 2*. This fact increases the author attribution difficulty on *dataset 2*. Also, we notice that *dataset 1* and *dataset 2* have similar results of the top 10 most frequent character 2-grams and character 3-grams. The main reason is that the #anonops channel contributes more active users than other channels for both datasets.

Rank	Dataset 1			
	Character 2-grams	Character 3-grams	Word 2-grams	Word 3-grams
1	“e_”	“_th”	“it’s”	“i don’t
2	“_t”	“the”	“don’t”	“www.youtube.com”
3	“t_”	“he_”	“i’m”	“youtube.com/watch”
4	“s_”	“ing”	“in the”	“https://www.youtube”
5	“th”	“ng_”	“of the”	“com/watch?v”
6	“_a”	“_to”	“if you”	“it’s not”
7	“_i”	“you”	“that’s”	“you don’t”
8	“in”	“_yo”	“you can”	“a lot of”
9	“he”	“to_”	“https://www”	“it’s a”
10	“d_”	“_an”	“to be”	“I’m not”

Table 4.12: The Top 10 Most Frequent Character N-grams and Word N-grams in Dataset 1

Rank	Dataset 2			
	Character 2-grams	Character 3-grams	Word 2-grams	Word 3-grams
1	“e_”	“_th”	“it’s”	“www.youtube.com”
2	“_t”	“the”	“i’m”	if if if
3	“t_”	“ing”	“don’t”	“https://www.youtube”
4	“s_”	“he_”	“in the”	“youtube.com/watch”
5	“_a”	“ng_”	“is a”	“com/watch?v”
6	“th”	“you”	“of the”	“i don’t”
7	“_i”	“_to”	“if you”	“friday it’s
8	“in”	“_yo”	“https://www”	“it’s flood”
9	“he”	“_an”	“i have”	“s flood friday”
10	“d_”	“_a_”	“youtube.com”	“flood friday it”

Table 4.13: The Top 10 Most Frequent Character N-grams and Word N-grams in Dataset 2

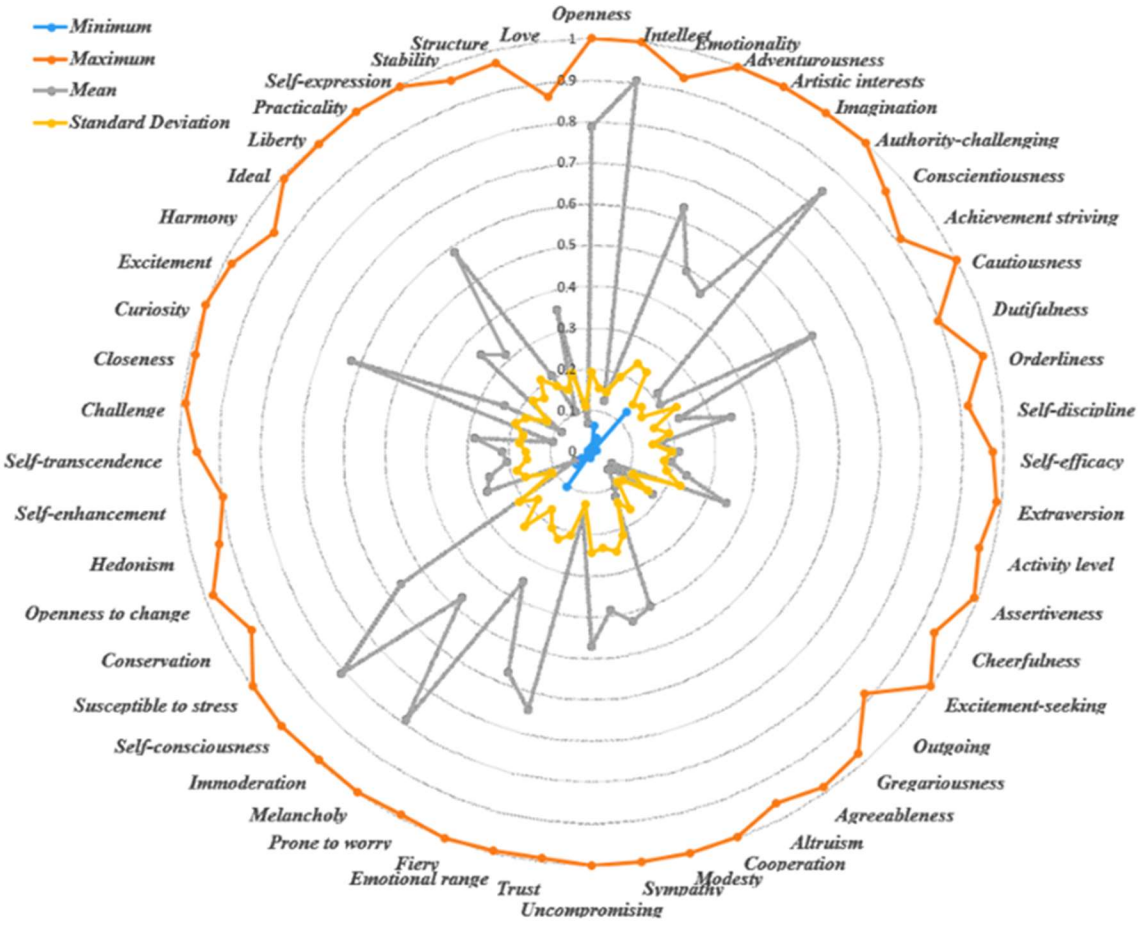


Figure 4.6: Statistical results of personality insights features from 4410 writing samples used in the experiments.

4.3.3 Computational Complexity Analysis

The first priority of most machine learning approaches is to create models that maximize accuracy. As models with equivalent prediction performance, a secondary objective is usually to minimize time complexity. Therefore, we also provide time complexity analysis of the attribution model learning phase (without time complexity analysis of FS unit). The time complexity of building a tree in the worst case is respectively $O(\sqrt{d}n^2 logn)$ and

$O(\sqrt{d}n^2)$ for a tree of RFs and a tree of ERTs, when given d features and n samples and setting the square root of the number of features are randomly drawn at each node [88]. Hence, in our approach, for a given training dataset $X = \{x_i, y_i\}(x_i \in \mathbb{R}^{d'}, y_i \in Y, |Y| = M, i = 1, 2, 3, \cdots, n)$, the time complexity for building a tree in the worst case is $O(\sqrt{d'}n^2 logn)$ in the CFS, where d' is the total number of features after applying FS unit, n is the total number of samples. Hence, the time complexity of processing each layer of CFS unit in the worst case is $O(w\sqrt{d'}n^2 logn)$, where w is the total number of trees built during the processing of each layer of CFS. Hence, the time complexity for the attribution model learning phase of AADF is $O(Lw\sqrt{d'}n^2 logn)$ in the worst case, where L is the total number of generated layers during the growth of CFS. Please note that AADF adaptively decides L in a data-dependent way for different tasks. In our experiments, the values of L are smaller than 4 in all the cases of dataset 1 and smaller than 5 in all the cases of dataset 2. The Scikit-Learn implementation of RF and ERT relies on NumPy vector operation and Cython for efficient numerical computing [83], which helps AADF to achieve scalability. Thus, AADF has acceptable computational performance and scalability. In addition, the training time can be further reduced if bootstrap samples are used.

We should note that the worst case of AADF happens in the very exceptional case when the splits are very unbalanced, which results in splits that move one sample in the first sub-tree and move all $n - 1$ other samples in the second sub-tree during all the decision trees constructions. Due to the fact that the decision tree looks for splits that further decrease impurity, the frequency of balance partitions of the nodes samples should be higher than partitions that are very unbalanced. Therefore, trees are more likely to approach the case when node samples are partitioned into balanced subsets, than very unbalanced subsets resulting in the average time complexity of building one tree is respectively $\Theta(\sqrt{d'}nlog^2 n)$ and $\Theta(\sqrt{d'}nlogn)$ for RF and ERT [88]. Thus, the time complexity of author attribution

Approach	Time Complexity
MNB	$O(d'n)$
ANN	$O\left(MI_{ANN}nd'\prod_{H=1}^{b}a_H\right)$
RF	$O\left(t\sqrt{d'}n^2 logn\right)$
SVM	$O(I_{SVM}nd')$
AADF	$O\left(Lw\sqrt{d'}n^2 logn\right)$

* for ANN, a_H is number of neurons in hidden layer H, b is total number of the hidden layers, and I_{ANN} is number of iterations
** t is number of tree in RF approach
*** I_{SVM} is number of iteration of SVM.

Table 4.14: The Worst Time Complexity of Attribution Model Learning Phase.

model learning phase of AADF is closer to $\Theta\left(Lw\sqrt{d'}nlog^2 n\right)$, resulting in good computational performance and scalability.

The worst time complexity of each approach is listed in Table 4.14. For the ANN, the time complexity is dominated by backpropagation. The worst time complexity of backpropagation is $O\left(MI_{ANN}nd'\prod_{H=1}^{b}a_H\right)$ [83], which is generally quite high and may perform many iterations for convergence. With increasing the number of layers and neurons and iterations, the training time of ANN is increased significantly. For the SVM, the time complexity in the worst case is $O(I_{SVM}nd')$ [85], the optimization problem task of the dual problem of SVM can be efficiently solved by a sequential minimal optimization-type decomposition method. It is empirically known that the number of iterations of performing decomposition method may be higher than linear to the number of training samples, resulting in considerable training time for huge datasets [85]. In our cases, the time complexity of SVM is lower than AADF. The MNB approach has a time complexity of $O(d'n)$ [89], which is the best among all the approaches in this study. MNB is simple and has an efficient computational nature due to the independence assumption of the features, though this assumption is always violated.

4.4 Summary

The anonymity of Internet services, especially in social media platforms, provides users with freedom. However, such anonymity can be exploited by cyber threat actors. Thus, it is critically important to develop methods to identify anonymous users involved in cybercrimes. In this chapter, to address this social media forensics challenge, we presented new ensemble learning-based author attribution techniques for IRC to accurately identify candidate authors from a large number of candidate sets. In our approach, we use the first author attribution version of deep forest to adaptively learn novel IRC writing features to identify IRC users. Unlike previous author attribution research in IRC who suffered from low accuracy given many candidates, our approach achieved more than 90% author attribution accuracy provided hundreds of candidates.

5. One-class Classification-based Author Verification Framework

5.1 Introduction

The most common social media forensics framework for identifying users is the author attribution problem that can be defined as attributing text samples of unknown authorship to one of the suspects when given a set of candidate authors and their texts samples of known authorship [3]. This framework is suitable for social media forensic cases when a specific set of candidates can be provided according to certain restrictions, such as prior knowledge of specific facts or access to specific material. Another demanding issue is author verification which can be formulated as follows: Verifying if a new digital text of unknown authorship is written by that particular author when given only one candidate author who has a set of digital texts of known authorship [14, 41]. One-class classification approaches try to identify objects of a specific class amongst all objects by learning from training samples containing only the objects of that class [90, 91]. Therefore, one way to formulate author verification is treating it as a one-class classification problem where the text samples written by the candidate author represent normal (negative) samples and all text samples from other authors represent abnormal (positive) samples. Although there are some author attribution studies in IRC, the research of author verification in IRC channels is unexplored. In our past paper [14], we proposed the first text author verification use of autoencoder. This chapter extends it by using the feature extraction model proposed in Chapter 4.

5.2 Author Verification Approach

In this section, we present our IRC-based author verification approach. Figure 5.1 shows our IRC author verification approach architecture. First, using our autonomic IRC bots deployed in various IRC channels, mainly about hacking and cracking activities, we monitor the IRC channels and users' messages and behaviors. The IRC messages of author

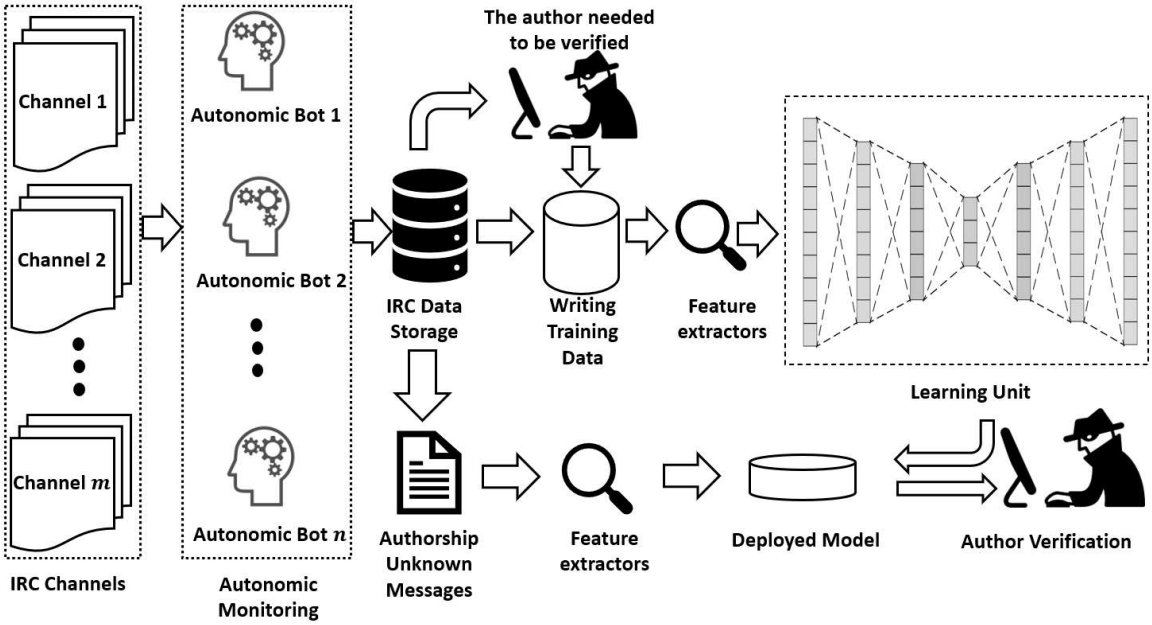


Figure 5.1: Overview of the author verification framework applied in IRC

candidates are extracted to perform two major steps for author verification: feature extraction and autoencoder-based author verification model construction. After completed the training of autoencoder, through autoencoder model, the extracted unknown authorship messages represented by extracted feature can be verified whether it belongs to a particular author.

5.2.1 Feature Extraction Model for Author Verification

We have demonstrated that our feature extraction model works well for the author attribution tasks. This feature extraction model can be directly applied to author verification in IRC. Therefore, we use the same feature extraction model used in Chapter 4 to extract features for autoencoder learning. Table 5.1 shows the description of the feature extraction model for author verification.

Feature Set	Description	The Number of Features
F1: Letters, numbers, special characters, punctuations	Frequency of letters, numbers, special characters, and punctuations	94
F2: Character n-grams	Frequency of the most 500 frequently ranked character 2-grams and the most 1,000 frequently ranked character 3-grams	1,500
F3: Word n-grams	Frequency of occurrence of the most 500 frequently ranked word 2-grams and the most 1,000 frequently ranked word 3-grams	1,500
F4: Stop-words	Frequency of stop-words	543
F5: Cybersecurity knowledge and attitude	Frequency of using NIST key information security terms	6,702
F6: Abbreviation	Frequency of IRC abbreviations	46
F7: Emoticons	Frequency of IRC emoticons	37
F8: Activity Behaviors	Amount of IRC messages in each half-hour	48
F9: Personality Insights	Big Five Model, Needs model, Values Model	52
F10: conversation network	Frequency of sending public one-to-one messages to a candidate	varies

Table 5.1: The feature extraction model for author verification in IRC

5.2.2 Author Verification with Autoencoder

An autoencoder is a feedforward network for learning a map from the input to itself through a pair of encoding and decoding phases [92]. A hidden layer in an autoencoder neural network generates a bottleneck enforcing autoencoder to compress input to a low-dimensional representation. For each input sample, we aim to train the autoencoder to make output and input as closely as possible. Autoencoder can be trained using the backpropagation algorithm with the output value to be equal to the input value. The forward propagation of a simple autoencoder with one hidden layer is given by:

$$h' = f\left(W^{(1)}x' + b_1\right) \tag{5.1}$$

$$y' = f\left(W^{(2)}h' + b_2\right) \tag{5.2}$$

where h' is the vector of representation of the hidden node activities and f is a non-linear activation function. $W^{(1)}$ is a parameter matrix between the input layer and hidden layer and b_1 is a vector of bias parameters. $W^{(2)}$ is a parameter matrix between the hidden layer and output layer and b_2 is a vector of bias parameters. x' is a vector representing input and y' is a vector representing the output. Autoencoder learns the parameters by performing gradient descent to minimize the reconstruction error. The mean squared error (MSE) is used to measure the reconstruction error in our approach. An autoencoder with higher model complexity can be achieved by using multiple hidden layers to learn the complicated distribution by given samples due to its multiple feature representation spaces. The mini-batch gradient descent algorithm with Adam optimizer [93] is used to train the autoencoder. Our autoencoder uses the ReLU activation function for each hidden layer and linear function for the output layer. The ReLU function is defined as follows:

$$f(x') = max(0, x') \tag{5.3}$$

The architecture of our autoencoder with symmetrical shape is shown in Figure 5.2. The input layer has 10,963 nodes since we extract features through our feature extraction model

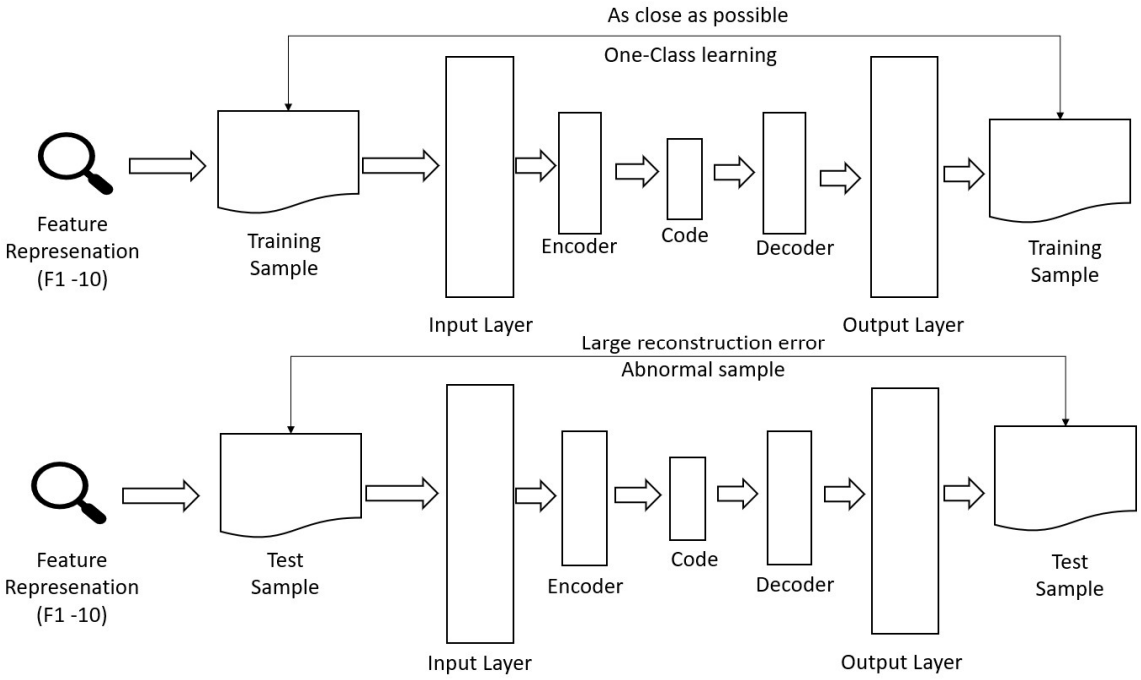


Figure 5.2: Overview of the autoencoder for author verification.

of IRC author identification described in Chapter 4. Therefore, the output layer also has 10,963 nodes since the purpose is to reconstruct the sample. Three fully connected layers (whose nodes are 128, 64, and 128, respectively) are added into autoencoder as hidden layers for forming an autoencoder. The middle hidden layer with the 64 nodes as the code layer which stores the compressed representation space for the input data. After completing the training stage, our autoencoder model can verify the authorship of IRC messages. Whether an IRC message sample belongs to a user is determined by whether the reconstruction error is higher than a threshold. In the test stage, an IRC message sample has a lower reconstruction error if the same author writes it. In contrast, the reconstruction error is larger if a different author writes it.

5.2.3 Author Verification with OC-SVM

One-class support vector machine (OC-SVM) is a one-class classification algorithm developed by Schölkopf et al. [100]. This algorithm map data into a feature space using a kernel function and try to separate the mapped features from the original space with maximum margin. We use the Radial Basis Function (RBF) that is given by:

$$K(x_i, x_j) = exp\left(-\gamma \left\| x_i - x_j \right\|^2\right) \tag{5.4}$$

Given the training data x_i, the algorithm can be obtained as follows:

$$\min_{w,\xi,\rho} \frac{1}{2} w^T w - \rho + \frac{1}{\upsilon m} \sum_{i=1}^{m} \xi_i \tag{5.5}$$
$$\text{Subject to } w^T \phi(x_i) \geq \rho - \xi_i \,,\ \xi_i \geq 0$$

where ρ represents the margin, w is a vector orthogonal to the hyperplane, and ξ_i are slack variables. υ represent an expected fraction of training samples that are allowed to be rejected, and m is the number of training samples of a particular author. The decision function is given by:

$$sign\left(\sum_{i=1}^{m} \alpha_i K(x_i, x) - \rho\right) \tag{5.6}$$

where the α_i is coefficients. The training samples x_i for which $\alpha_i \neq 0$ are treated as support vectors.

5.2.4 Author Verification with Isolation Forest

Isolation Forest (iForest) [101] is an isolation-based anomaly detection method that utilizes an ensemble of isolation trees. Each isolation tree is a decision tree created for isolating each sample from the rest samples in a given training data from the one-class author candidate (normal class) who needed to be verified.

The isolation tree measures individual samples' susceptibility to being isolated; With the hyper-parameter ψ (the size of random subsample), each partition of the isolation tree is performed by randomly selecting a split value between the maximum and minimum values of the selected features in the random subset of training data. The partitioning procedure is repeated recursively until each training sample is isolated from the rest of the training samples. iForest is an ensemble of a number of isolation trees. Each isolation tree is built by random subsample from the training data.

During the test phase, a sample having a short path to be isolated by isolation tree through a few partitions is more likely to be an abnormal sample. The anomaly score for each test sample is calculated using the average path length from all isolation trees.

5.3 Experiments and Results

5.3.1 Setup

To evaluate our approach, we performed author verification tasks on a dataset named IRC-AV(author verification). The collected data shown in Table 4.2 is used to generate the IRC-AV dataset. Table 5.2 described the IRC-AV datasets. More specifically, except without performing the undersampling step, we use the same strategy used to create the IRC dataset 1 in author attribution tasks (in Chapter 4) to create the IRC author verification experimental dataset. In this way, 441 different users (with 19,004 samples) are extracted. We ranked the potential malicious users according to the summation of the threat level score (normal=0, warning=1, and high=2) of all their own IRC messages. The top 30 potential malicious users are then selected as the target author who needs to be verified.

Dataset	Authors	Features	Samples per Authors	Words per Samples	Samples
IRC-AV	441	10,963	Unbalanced	1,000	19,004

Table 5.2: Description of experimental IRC author verification dataset.

Hence, the author verification experimental dataset has 441 different classes with 19,004 samples, where we create 30 one-class classification-based author verification experiments. In each experiment, one of the 30 target authors is treated as the normal (negative) class, and samples from the remaining 440 classes are represented as an abnormal (positive) class. We only use the samples from the respective normal class for the training set for all three anomaly detection models due to the one-class learning principle. Therefore, random sampling is used on normal data to extract 60% of normal samples for training. The remaining 40% of normal samples are mixed with all abnormal samples for testing. Random sampling is repeated 10 times to produce 10 rounds of different combinations of training set and test set for evaluation. Min-max normalization is used to normalize the range of features [83]. The area under the curve (AUC) [131] is used to measure the performance of author verification models. AUC summarizes the entire location of the receiver operating characteristic (ROC) curve, which shows the tradeoff between true positive rate and false positive rate for various thresholds [83].

5.3.2 Results

Table 5.3 shows the results of three verification models in terms of average AUC using the feature extraction model for author verification. In most cases of authors, they show good performance for one-class classification, addressing the effectiveness of our feature extraction models presented in Chapter 4.

We also notice that the overall performance of the autoencoder is superior to IF and OC-SVM. To be more specific, autoencoder achieved the best results on 21 authors, while iForest and OC-SVM approach best results on 9 and 10 authors, respectively.

Normal Class (Nickname)	Sum of threat level score	Total # of Samples	iForest	OC-SVM	Autoencoder
JARVIS	13126	370	**99.88** ± 0.03	99.68 ± 0.28	99.69 ± 0.29
Meow	12524	212	99.98 ± 0.01	**100** ± 0.00	**100** ± 0.00
Gopher	5006	592	97.13 ± 0.30	99.41 ± 0.36	**99.45** ± 0.36
thufir	4908	62	99.55 ± 0.13	**100** ± 0.00	**100** ± 0.00
aestetix	4213	513	98.26 ± 0.24	98.64 ± 0.27	**99.83** ± 0.11
Effexor	3703	610	**99.68** ± 0.14	97.47 ± 1.19	97.42 ± 0.12
zeta	3411	81	99.56 ± 0.24	**99.99** ± 0.01	**99.99** ± 0.01
piqure	3228	38	96.04 ± 2.89	**100** ± 0.00	**100** ± 0.00
WolfBot	3183	44	**99.95** ± 0.05	99.49 ± 1.51	99.49 ± 1.51
RDNt	3150	228	83.74 ± 1.68	86.19 ± 1.61	**89.46** ± 1.31
LostBiT	2973	128	99.96 ± 0.03	**100** ± 0.00	**100** ± 0.00
catphish	2700	173	97.64 ± 0.46	98.65 ± 0.61	**98.66** ± 0.61
keiththewhteguy	2587	253	98.38 ± 0.40	99.23 ± 0.56	**99.25** ± 0.56
druqs	2493	69	99.71 ± 0.09	99.99 ± 0.01	**100** ± 0.00
stovepipe	2451	374	96.28 ± 0.54	98.77 ± 0.54	**98.84** ± 0.51
shbot	2207	38	99.93 ± 0.04	**99.99** ± 0.01	**99.99** ± 0.01
covfefe	2047	245	94.84 ± 0.94	98.42 ± 0.66	**98.49** ± 0.64
djph	1997	135	**99.43** ± 0.16	99.09 ± 1.12	99.07 ± 1.12
greycat	1869	166	99.57 ± 0.05	**99.99** ± 0.01	**99.99** ± 0.01
maestro	1822	230	87.70 ± 1.17	97.81 ± 0.68	**98.02** ± 0.61
twelve	1791	188	**99.84** ± 0.15	99.57 ± 0.45	99.57 ± 0.45
greybot	1775	192	**100** ± 0.00	**100** ± 0.00	**100** ± 0.00

WildMan	1754	63	**99.80** ± 0.08	99.45 ± 0.90	99.43 ± 0.92
Cogitabundus	1738	222	**99.97** ± 0.03	99.84 ± 0.24	99.84 ± 0.24
Crono_	1724	402	99.74 ± 0.08	**99.79** ± 0.21	**99.79** ± 0.21
Cochise	1678	147	99.13 ± 0.92	99.22 ± 0.66	**99.25** ± 0.60
dmt	1589	207	97.25 ± 0.44	99.61 ± 0.12	**99.73** ± 0.08
catface	1572	165	86.55 ± 1.36	87.03 ± 1.34	**89.95** ± 1.21
mickers	1544	178	97.69 ± 0.41	**99.30** ± 0.45	99.28 ± 0.45
lazarus	1515	74	**100** ± 0.00	98.50 ± 1.88	98.41 ± 1.93

Table 5.3: Average AUC of Different Verification Models with Feature Sets F1-F10.

5.4 Summary

In this chapter, we investigate the author verification tasks for social media forensics. We address our author verification framework through its application to IRC. Although IRC is one of the most popular social media platforms leveraged by cyber threat actors, IRC author verification is an unexplored task. On the other hand, almost all the previous works for one-class classification-based text author verification use SVM-based approaches [40, 41]. In this thesis, we presented the first text author verification use of autoencoder and address its superiors through 30 IRC author cases of one-class classification experiments. Over 70% of author cases in the experiments, our author verification approach can achieve more than 99% AUC.

6. Unsupervised Learning-based Author Clustering Framework

6.1 Introduction

Most previous author identification studies focus on identifying authors of unknown texts when given a set of author candidates with their text samples of known authorship [20]. However, there are cases when the given author information is unavailable [48]. For example, in a collection of proclamations written by anonymous groups, we may not have clear suspects and their known writing to learn the identification model. Without training authorship, supervised author identification techniques are hard to be designed for identifying authorship. This fact suggests the requirement for developing author identification techniques to identify the authors using unlabeled data through unsupervised learning. Author clustering is a promising approach to tackle the author identification problem among unlabeled data, which can be formulated as follows: Grouping a set of digital texts of unknown authorship written by their author [15]. However, the research of author clustering is very limited, especially for the Twitter platform.

Twitter is one of the most popular social media platforms around the world [94]. Twitter allows users to disseminate their opinions, share information, and chat with people across the globe immediately. Although Twitter usage has evident advantages in social media, it also raises cyber threats and cyber-crimes such as stolen confidential data trade, hacking tools propagation, phishing attacks, and misinformation propagation [7, 33]. Furthermore, recent studies have found that: organized trolling campaigns turned into a new phase, whereby governments and corporations make effort to influence the opinion surrounding important events (both real and fake) on Twitter [11, 13]. For example, the U.S. Justice Department indicated a Russian state-sponsored media agency named the Internet Research Agency (IRA) to disseminate discord in the US political system [10]. The agency employed hundreds of people to create fake accounts on social media to promote the Russian government's interests in domestic and foreign policy, including Ukraine and the

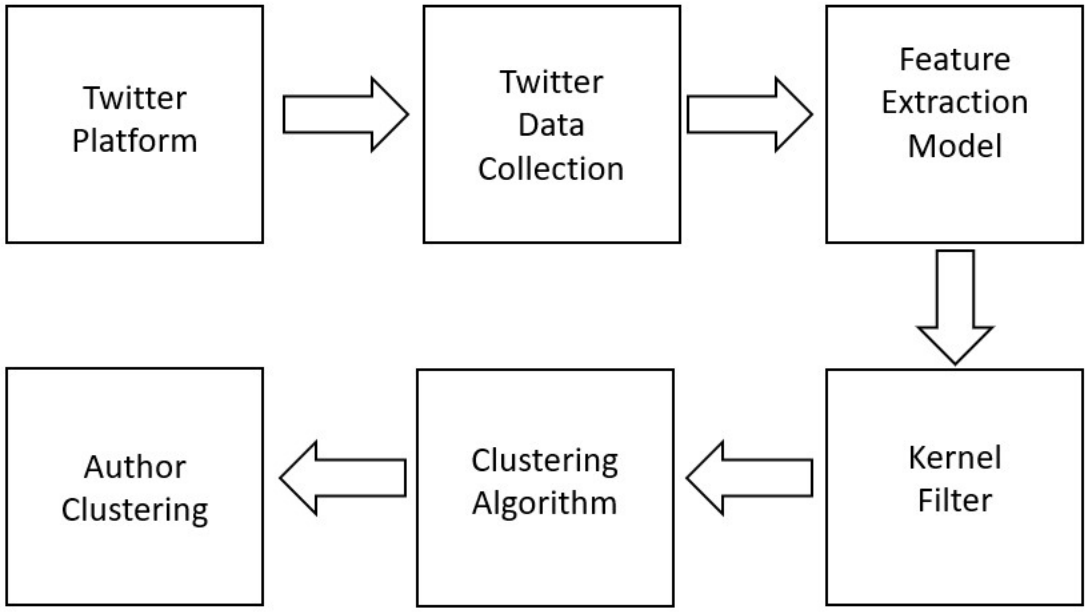


Figure 6.1: Architecture of Twitter Author Clustering.

Middle East, and attempting to influence the 2016 US presidential election [10]. Then, Twitter identified 3,814 IRA-linked accounts masqueraded as US citizens to divide voters into a series of issues, such as the Black Lives Matter movement, immigration, and feminism [10]. Furthermore, cybercriminals can rely on anonymity methods to ensure the success of cyber-crime [3]. Hence, the effective method for monitoring, analysis, and identification has become critically important. For this purpose, this chapter proposes a novel author clustering framework and its application to Twitter.

6.2 Twitter Author Clustering Framework

Figure 6.1 presents our author clustering framework for Twitter. First, we collect cyber threat actor data using our automatic Twitter data collection and detection tool (described in Chapter 3). To generate samples representing their writing behaviors, we use our proposed feature extraction model to cyber threat actors' preprocessed tweets. In the author clustering phase, a liner kernel filter converts high dimensional data to a kernel matrix for the EM-based GMM clustering that can effectively group samples by their author.

6.2.1 Feature Extraction Model for Author Clustering

The author identification feature extraction model proposed in Chapter 4 (shown in Table 4.1) can also be adapted to Twitter for author identification analysis with several simple changes. Due to their generalization, the feature sets F1, F2, F3, F4, F5, and F9 can be applied to author identification in other platforms. The remaining feature sets (i.e., F6, F7, F8, and F10) provide a high-level concept of feature extraction methods for other social media applications, though platform-specific feature extraction might be required. Therefore, for Twitter author clustering, the feature sets F1, F4, F5, and F9 are directly applied, the feature sets F2 and F3 are applied with optimized parameter settings (i.e., the values of n of character n-gram and word n-gram and their frequency ranked cutoff points). The feature sets F6, F7, F8, and F10 are adapted to the content of Twitter. A feature set, F11, is designed to leverage the hashtags from Twitter. Table 6.1 summarizes the feature extraction model for Twitter author identification tasks. Applying the feature sets F1, F2, F3, F4, F5, and F9 from our proposed IRC feature extraction model to Twitter author identification is straightforward. Therefore, we only describe the feature sets of F6, F7, F8, F10, and F11 in this chapter.

Twitter Abbreviations: We investigated special content features that appeared on Twitter in terms of the Twitter common abbreviation. Twitter contains abbreviations like “MT” (modified tweet), “TMB” (tweet me back), “PRT” (please retweet), etc. We used 132 non-repeating abbreviations from the study [95] to represent Twitter user’s preference for writing abbreviations.

Twitter Emoji: Twitter emojis are used to express emotions. Emojis exist in different genres such as facial expressions, animals, various objects, weather, country, etc. We adopted the most top 200 popular Twitter emojis from the emoji tracker website [96]. Thus, the writing characteristics of users can be analyzed through an emoji feature set.

Feature Set	Description	The Number of Features
F1: Letters, numbers, special characters, punctuations	Frequency of letters, numbers, special characters, and punctuations	94
F2: Character n-grams	Frequency of the most 2,000 frequently ranked character 2-grams, the most 2,000 frequently ranked character 3-grams, and the most 2,000 frequently ranked character 4-grams.	6,000
F3: Word n-grams	Frequency of occurrence of the most 500 frequently ranked word 2-grams.	500
F4: Stop-words	Frequency of stop-words	543
F5: Cybersecurity knowledge and attitude	Frequency of using NIST key information security terms	6,702
F6: Abbreviation	Frequency of Twitter abbreviations	132
F7: Emoji	Frequency of Twitter Emoji	200
F8: Activity Behaviors	Amount of Tweets in each half-hour	48
F9: Personality Insights	Big Five Model, Needs model, Values Model	52
F10: Twitter Mentions and Replies	Frequency of frequently occurred *@username*	varies
F11: Twitter Hashtags	Frequency of frequently occurred hashtags	varies

Table 6.1: The Feature Extraction Model for Author Identification in Twitter.

Twitter Hashtags: Hashtags can be used to simplify content searching on Twitter. Hence, cybercriminals can use hacking words as hashtags. For extracting hashtags features, we use the frequency cut-off approach to reduce the high dimensionality of data. Any hashtag that appeared at least ten times in the given dataset is extracted as a hashtag feature.

Twitter Mentions and Replies: While some malicious users are interested in and experienced in discovering vulnerabilities, some focus on diffusing anonymous organization cultures and recruiting new members. Therefore, by analyzing replies and mentions, we can create representation of users' connection and relevance. To reduce the high dimensionality of data, we use the frequency value cut-off approach. We select the appeared @username feature if it appeared at least ten times in the given dataset.

Twitter Activity Behaviors. There exist multiple factors impacting activity behavior, such as the location and tweets post habits. For Twitter, we extract time slot features where a day 48 half-hours are split into 48 equal-size intervals. Then, we count the number of tweets that have been posted in each half-hour interval.

6.2.2 Unsupervised Author Clustering

To improve the performance of clustering models, we use a kernel filter to preprocess the dataset. Kernel filter is a preprocessing filter without using label information. Kernel filter converts N' dimensions data set into a kernel matrix through applying a kernel function to each pair of samples in the dataset. The new features in the transformed dataset hold the kernel evaluation between a pair of instances, thus generating an $n \times n$ matrix (n is the number of samples). The kernel matrix is given by:

$$\begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_n) \\ K(x_2, x_1) & K(x_2, x_2) & & \\ \vdots & & \ddots & \vdots \\ K(x_n, x_1) & & \cdots & K(x_n, x_n) \end{bmatrix} \quad (6.1)$$

where $K(x_i, x_j)$ is kernel function expressed as inner products of two samples. Considering the very large numbers of features that appeared in our dataset, we use the polynomial kernel function of degree 1, which is equivalent to the linear kernel. Hence, the dataset X is transformed to E by mapping into a feature space by apply kernel filter. The dimensionality of the dataset is reduced from d to n.

Gaussian Mixture Models (GMM) is a mixture density models assuming that each component of the probabilistic model is a Gaussian density as followers:

$$p(e|\theta) = \sum_{i=1}^{M} \alpha_i G_i(e|\theta_i) \qquad (6.2)$$

where $e \in \mathbb{R}^n$ is the sample of clustering dataset after applying kernel filter, $\Theta = (\alpha_1, \cdots, \alpha_{M-1}, \alpha_M; \theta_1, \cdots, \theta_{M-1}, \theta_M)$ satisfy $\sum_{i=1}^{M} \alpha_i = 1, \alpha_i \geq 0$. Gaussian probability density $G_i(e|\theta_i)$ is given as:

$$G_\lambda(e|\theta_\lambda) = \frac{1}{(2\pi)^{n/2}|\sum_\lambda|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_\lambda)^{\mathrm{T}} \sum_\lambda^{-1} (e-\mu_\lambda)\right\} \qquad (6.3)$$

where $\theta_\lambda = (\mu_\lambda, \sum_\lambda)$. With introducing the latent variable item $Z = \{z_i\}_{i=1}^n$ whose value indicates which component generates the data, GMM assumes the data set $E = \{e_i\}_{i=1}^n$ are generated through M Gaussian components. Hence, $z_i = \lambda$ if a text sample is generated by the λ^{th} component. Hence, GMM's parameters can be estimated by the EM algorithm [97]. EM is an iterative procedure algorithm estimating the new parameters in terms of the old parameters. The updating formulas are given as:

$$\alpha_\lambda^{(t)} = \frac{1}{n} \sum_{i=1}^{n} p\left(l \middle| e_i, \Theta^{(t-1)}\right)$$

$$\mu_\lambda^{(t)} = \frac{\sum_{i=1}^{n} e_i p\left(l \middle| e_i, \Theta^{(t-1)}\right)}{\sum_{i=1}^{n} p\left(l \middle| e_i, \Theta^{(t-1)}\right)}$$

$$\sum\nolimits_l^{(t)} = \frac{\sum_{i=1}^{n} \left(e_i - \mu_\lambda{}^{(t)}\right)\left(e_i - \mu_\lambda{}^{(t)}\right)^T p\left(\lambda \middle| e_i, \Theta^{(t-1)}\right)}{\sum_{i=1}^{n} p\left(\lambda \middle| e_i, \Theta^{(t-1)}\right)} \qquad (6.4)$$

Step 1: Due to $d \gg n$, the very high dimensional dataset is mapped into a lower-dimensional feature space by apply linear kernel filter.
Step 2: Initialize the number of clusters as 1.
Step 3: Dataset is randomly split into 10 folds.
Step 4: EM-based GMM is performed by 10-fold cross-validation for calculating the average log-likelihood overall 10 fold.
Step 5: If the average log-likelihood is increased, the number of clusters is added by 1, and the program execution continues from step 3; otherwise, output the current number of clusters as the final number of clusters.

Table 6.2: Steps of estimating the number of authors in a given unlabelled data.

where λ means the λ^{th} component of GMM, and

$$p\left(\lambda\middle|e_i, \Theta^{(t-1)}\right) = \frac{\alpha_\lambda^{(t-1)} G\left(e_i\middle|\theta_\lambda^{(t-1)}\right)}{\sum_{j=1}^{M} \alpha_j^{(t-1)} G\left(e_i\middle|\theta_\lambda^{(t-1)}\right)}, \lambda = 1, \cdots, M \qquad (6.5)$$

where $\Theta^{(t-1)} = \left(\alpha_1^{(t-1)}, \cdots, \alpha_{M-1}^{(t-1)}, \alpha_M^{(t-1)}; \theta_1^{(t-1)}, \cdots, \theta_M^{(t-1)}\right)$ and $\Theta^{(t)} = \left(\alpha_1^{(t)}, \cdots, \alpha_{M-1}^{(t)}, \alpha_M^{(t)}; \theta_1^{(t)}, \cdots, \theta_{M-1}^{(t)}, \theta_M^{(t)}\right)$ are parameters of the $(t-1)^{th}$ iteration and $(t)^{th}$ iteration, respectively.

To estimate the number of authors in a given unlabelled data, we describe the steps (see Table 6.2) to estimate the number of authors in a given unlabelled high dimensional data.

6.3 Experiments and Results

6.3.1 Setup

To evaluate our method, we performed Twitter author clustering in two cases, clustering Twitter aliases into (1) *known* and (2) an *unknown* number of authors. The former simulates the social media forensic case when the investigator knows the number of authors (who create multiple alias accounts), and alias accounts should be grouped by authorship. The latter simulates the social media forensic case when the number of authors (who create multiple alias accounts) should be identified, and alias accounts should be grouped by authorship.

A new corpus of potential malicious users was developed for author clustering experiments using the data collected by our automatic Twitter data collection tool. One hundred twenty distinct Twitter users related to hacker groups are extracted for creating six experimental datasets. Each dataset has 20 distinct users. The description of the six experimental datasets is shown in Table 6.3. Then we perform non-overlapped segments to each Twitter user's whole tweets to a series of samples containing 1000 equal words since 1,000 words is regarded as a suitable and reliable choice for author identification [79]. Thus, each sample represented by the features from the feature extraction model is used to simulate and represent an alias account of a user (we discarded the remaining tweets that are less than 1,000 words). For each dataset, we use 70% samples from 20 authors as training set, and the remaining 30% from 20 authors as test set. We apply the min-max normalization to normalize the range of features. Since our task is author clustering, the goal of the training set is to select the values of the hyperparameters used in the test set. After finishing the tuning of our approach on the training set, we use the test set to evaluate the final performance.

Twitter Dataset #	Total # of authors	Total # of samples in dataset	Word count per sample
Twitter-1	20	586	1,000
Twitter-2	20	550	1,000
Twitter-3	20	735	1,000
Twitter-4	20	546	1,000
Twitter-5	20	742	1,000
Twitter-6	20	869	1,000

Table 6.3: Description of Twitter author clustering datasets.

To understand the complexity of author clustering tasks and the effectiveness of our approach, we used a number of baseline clustering models, including K-Means Clustering, Hierarchical Agglomerative Clustering (HAC), and X-Means Clustering (extending K-means that can estimate the number of clusters).

We use classes-to-clusters evaluation, a clustering evaluation measure developed by Weka team [98]. The advantage of this clustering evaluation measure is that it treats whether appropriately estimate the number of clusters is an important factor. In social media forensic tasks, knowing the scale of participants (authors) is critically important, especially for organized cybercrime and trolling campaigns. The procedure of classes-to-clusters evaluation is described as follows: After the execution of clustering algorithm is completed, for evaluating the performance of clustering, the classes-to-clusters evaluation uses a brute-force approach to search the minimum error assignment of class labels to clusters with a constraint that one particular class label can only be assigned to one cluster. The clusters without giving class label assignments receive "*No class*". Suppose there is a situation where the error is equal between the assignment of one particular class to one of several clusters. In that case, the first cluster considered during the search receives the assignment.

EM-GMM	EM-GMM (kernel filter)	K-Means	K-Means (kernel filter)	HAC	HAC (kernel filter)
76.07	**81.93**	68.40	65.49	17.26	66.10

Table 6.4: The average accuracy of each approach with knowing the number of authors.

This mapping is then used to assign class labels for samples. Note that the label is not used in clustering, while it is used to evaluate the clustering. All the samples are treated as misclassified if they are clustered to "*No class*". Accuracy is used to measure the performance of approaches.

6.3.2 Results

Clustering Aliases with Prior Knowledge of the Number of Authors. The average accuracy of each approach is shown in Table 6.4. We notice that EM-based GMM (with kernel filter) achieves the best results. We also observed that kernel filter significantly improved the performance of EM-based GMM and hierarchical agglomerative clustering. The Accuracy of different approaches on each experimental dataset is shown in Figure 6.2. As we can see, EM-based GMM (kernel filter) achieved the highest accuracy in four datasets, including Twitter-1, Twitter-2, Twitter-3, and Twitter-5, while EM-based GMM (without kernel filter) achieved the best accuracy in two datasets, including Twitter-4 and Twitter-6.

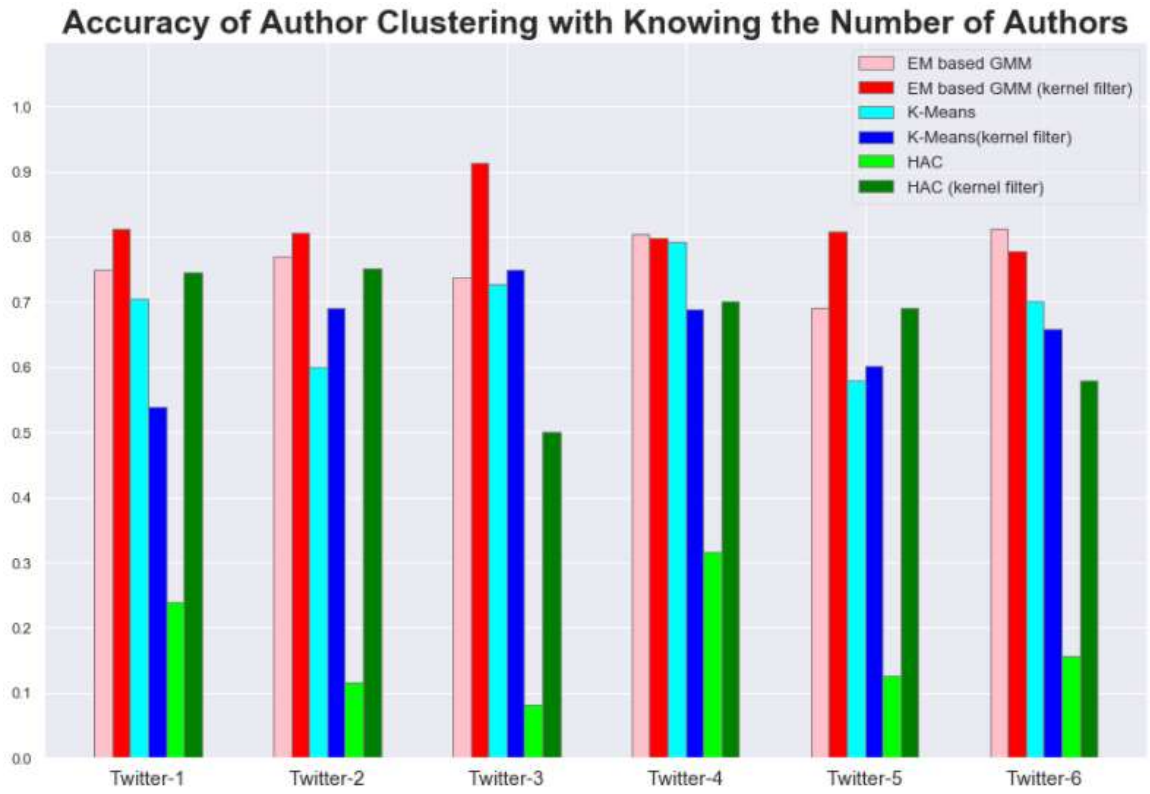


Figure 6.2: Accuracy of different approaches on each experimental dataset for author clustering task with knowing the number of authors.

Clustering Aliases without Prior Knowledge of the Number of Authors. EM-based GMM and *X*-Means Clustering are considered, and each clustering algorithm is performed with or without kernel filter. An important decision for such a case is appropriately estimating the number of authors (clusters) in given data. For the EM-based GMM, the method of estimating the number of clusters is described in Table 6.5. For *X*-Means, it can estimate the number of clusters through BIC [99].

The average accuracy of each approach is shown in Table 6.6. We notice that EM-based GMM (with kernel filter) achieves the best results. The Numbers of clusters detected by different clustering models with/without using kernel filter are presented in Table 6.5. Number of authors (δ) for each dataset are 20 since we use the same datasets from the previous experiments. ζ_1, ζ_2, ζ_3, ζ_4 are represented as the number of detected clusters of EM-based GMM with kernel filter, EM-based GMM without kernel filter, *X*-Means with

Twitter Dataset #	δ	ζ_1	ζ_2	ζ_3	ζ_4
Twitter-1	20	20	5	22	4
Twitter-2	20	13	4	8	2
Twitter-3	20	16	6	27	2
Twitter-4	20	19	5	20	3
Twitter-5	20	15	8	40	4
Twitter-6	20	23	13	37	9

Table 6.5: Number of authors and detected clusters.

EM-GMM	EM-GMM (kernel filter)	X-Means	X-Means (kernel filter)
44.17	**74.78**	26.45	73.95

Table 6.6: The average accuracy of each approach without knowing the number of authors.

kernel filter, and *X*-Means without kernel filter, respectively. We observed that kernel filter significantly improved the self-estimated capability of clustering models when estimating the number of clusters (authors) among a large number of unlabeled text samples in the datasets. More specifically, EM-based GMM (kernel filter) accurately estimated the number of clusters in the dataset of Twitter-1, while *X*-Means (kernel filter) accurately determined the number of clusters in the dataset of Twitter-4. As for the other datasets, the numbers of detected clusters of EM-based GMM (kernel filter) are closer to the number of authors than other methods.

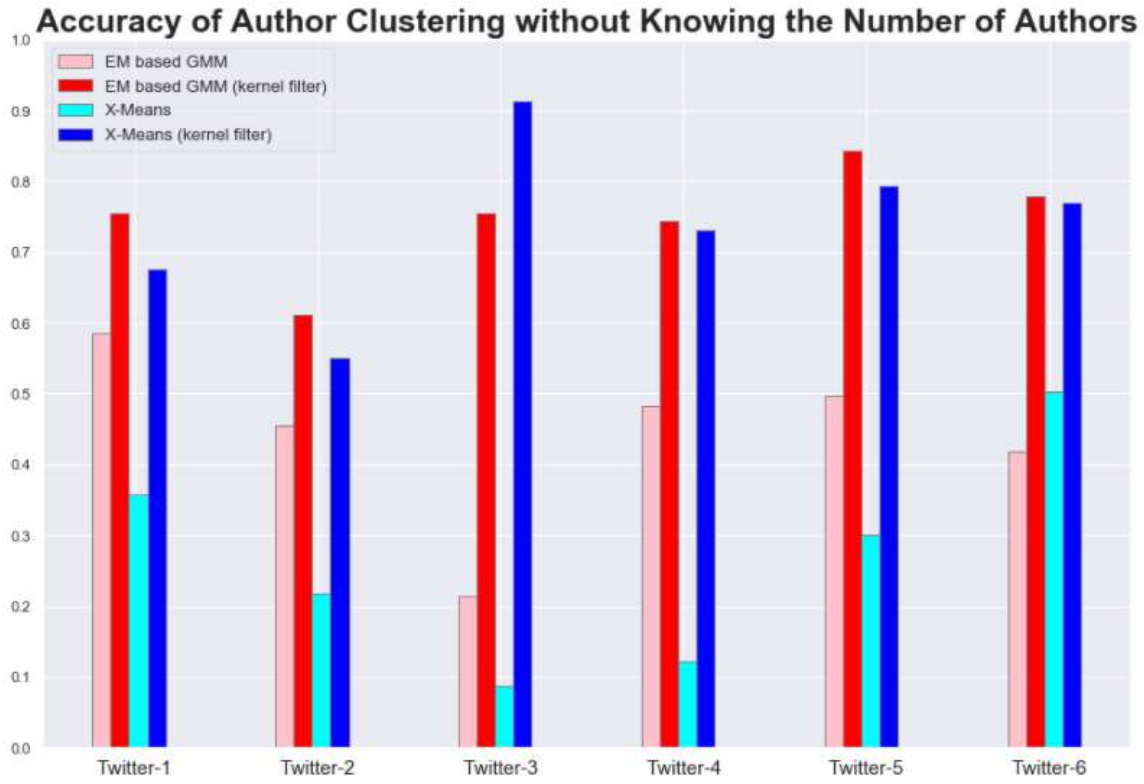


Figure 6.3: Accuracy of different approaches on each experimental dataset for author clustering without knowing the number of authors.

Figure 6.3 present the results of different approaches on each experimental dataset for author clustering without knowing the number of authors. We observe that kernel filter consistently and significantly improve clustering models. Further, EM-based GMM (kernel filter) outperforms other methods on all the datasets except the dataset of Twitter-3.

6.4 Summary

The Twitter platform provides freedom of allowing each user to control many accounts, resulting in an increasing trend that cyber threat actors sign up many aliases for malicious behaviors. Hence, it is highly desired to design techniques that can group alias accounts by authorship. This chapter has developed an automated unsupervised learning approach for Twitter author clustering to address the cybersecurity challenge. We first used our

automatic Twitter data collection tool to gather potential malicious user data. After that, we perform feature extraction, then convert high dimensional data to kernel matrix for Twitter author clustering. Compared to the previous work for Twitter author clustering, our approach can effectively identify the groups among many aliases even without knowing the number of authors. In the experiments given more than one hundred unlabeled text samples, our author clustering approach attains an average accuracy of 81.93% when knowing the number of authors and an average accuracy of 74.78% without prior knowledge of the number of authors in the given datasets.

7. Conclusions

Identifying and unmasking the authorship information is essential to reduce misuse, predict cybercrime, enhance credibility, and assist and enable text forensic investigation of social media platforms. This thesis presented three novel author identification frameworks for social media forensics and enabling systematic ways to identify users for digital forensic analysis under different author identification tasks. This thesis attempts to fill the research gaps for author identification in Twitter and IRC and advance research of author identification in social media forensics. Therefore, we developed novel author identification frameworks to collect and detect digital text, extract the holistic writing feature, and effectively learn these features through machine learning-based methodologies. We addressed the effectiveness and demonstrated the feasibility of our frameworks through a series of experiments in different author identification tasks. This chapter summarizes the contributions and broad impacts and then discusses potentially future directions.

7.1 Contributions of this Work

Automatic data collection and threat detection tools. An autonomic IRC monitoring tool was developed. In addition to collecting comprehensive data for monitored channels, it can classify threats as a result of a recursive deep model and intelligently simulate interaction with users in the channels. We also developed an automated Twitter user data collection and threat detection tool that can reslove several major limitations of Twitter API and detect threats.

Feature extraction model for author identification. We developed an author identification feature extraction model that incorporates holistic writing behavior features that can easily be customized for various social media platforms for different author identification tasks, including author attribution, author verification, and author clustering.

Author attribution deep forest. We developed the first author attribution version of the deep forest (DF) model, an ensemble of ensembles with forest-based feature selection, fewer hyper-parameters, and automatically determine the model complexity through a data-dependent manner. Our author attribution experimental results show that its attribution capability is very robust to the settings of hyperparameters. It outperformed state-of-the-art models in author attribution, including MLP-based ANN, SVM, RF, and MNB, across all of our author attribution experiments, even using the same setting of hyperparameters in the CFS structure.

A novel author verification framework. A novel author verification framework under the principle of one-class learning was developed that presented the first text author verification use of the autoencoder according to the reconstruction-based one-class classification concept.

A new author clustering framework. A new unsupervised author clustering framework was developed. It combines kernel filter, EM, GMM, and log-likelihood-based cross-validation. The experimental results show that it can group unknown author writings even without prior knowledge of the number of authors.

7.2 Broader Impacts

Although we demonstrated that our frameworks work well in IRC and Twitter, our author identification approaches, including author attribution, author verification, and author clustering, can also be applied to other social media platforms for different author identification tasks. In the feature extraction phase, applying feature sets F1, F2, F3, F4, F5, and F9 to forensic investigations of other platforms is straightforward due to their generalization. The remaining feature sets (i.e., F6, F7, F8, and F10) provide a high-level concept of feature extraction methods for other social media applications, though platform-specific feature extraction might be required.

The AADF ensemble model achieved the best performance in all cases in our attribution experiments in the same setting. Hence, we believe that our ensemble learning-based model

would perform well for author attribution tasks across many social media platforms because ensemble classifier usually provides better classification results than one-classifier. The AADF can be viewed as an ensemble of ensembles that leverage almost all strategies for diversity enhancement. The AADF can also provide benefits to other authorship analysis tasks such as author profiling [102]. The RF, SVM, and MNB are widely utilized for author profiling [103]. Hence, our ensemble of ensembles-based authorship analysis approach provides an opportunity to improve the accuracy due to its best performance in this study, which also falls in the area of the high-dimensional text classification problem.

The autoencoder for IRC author verification can be easily adapted to the author verification tasks in other social media platforms under the reconstruction-based one-class classification concept. Our author clustering framework is also capable of author clustering tasks in a broad range of social media platforms.

Additionally, the author clustering approach is applicable to other unsupervised authorship analysis tasks such as style change detection [127], whose mission is to identify the positions in the text within a given multi-author text where the author changes.

7.3 Future Work

This thesis touched on many elements of author identification. However, many interesting open questions still remain, which can be explored in future research.

7.3.1 Dealing with Data Quality Uncertainty in Author Identification

Data quality plays a critical role in machine learning-based author identification, particularly in author attribution due to its requirement of ground-truth labels to be given for training. It is noteworthy that it might be difficult to ensure high data quality for proper supervision in some author attribution tasks because reliable training data are tedious and time-consuming to collect for some author candidates. In contrast to the well-supervised author data, weakly supervised author data has less quality of label information, resulting

in the label information can be either coarse-grained or wrongly labeled. Therefore, it would be desirable for author attribution techniques to deal with weakly supervised data effectively.

Multi-instance learning [132] is a promising technique that can improve performance when incorporating coarse-grained label information in the author's data. The work of Leistner et al. [120] proposed a variant of random forest that can effectively perform multi-instance learning. Label noise learning [133] technique is one way to learn a promising prediction from the noise supervised data whose labels are not always been ground-truth. In [121], a two-stage method to detect label noise based on the random forest has shown effectiveness for label noise learning. Therefore, the adoption of random forest as the base learner in AADF not only helps enhance diversity but also provides the potential capability for weakly supervised learning for dealing with data quality uncertainty in author identification.

7.3.2 Author Identification Against Author Obfuscation

In author obfuscation, the objective is to make author identification impossible or at least difficult [122]. An interesting study [104] developed three disguised writing methods for circumventing authorship recognition: obfuscation, imitation, and machine translation. Their results showed that manually disguised writing methods, including obfuscation and imitation, work well, whereas automated translation is ineffective. One promising way to improve the success rate against disguised writing methods is to collect long-term data of suspects since successful circumventing authorship analysis is limited to manual methods that require hand-crafted manipulation [105] or semi-auto methods that provide possible modification suggestion [106], and using these methods consistently is even hard for sophisticated hackers.

Moreover, the adoption of non-traditional feature sets, such as personality insight, may reveal the adversarial authors who attempt to circumvent author identification systems by contaminating traditional writing features documented in the literature.

It should be noted that our author identification approaches are unlikely to maintain performance if suspects intentionally disseminate their writing style for contaminating statistics. Therefore, one important future issue is to enhance the robustness of author identification techniques against author obfuscation.

7.3.3 Detecting Adversarial Examples for Author Identification

MLP neural networks and autoencoder neural networks achieved good author attribution and author verification performances in this research. However, neural networks are susceptible to adversarial examples such as inputs similar to a correctly classified input, but misclassified [123]. This has led to extensive studies on the use of maliciously crafted adversarial examples to attack neural networks [126], providing another possible option for attacking neural networks-based author identification models. Therefore, defenses for neural networks-based author identification model is one future direction.

When creating adversarial examples, most methods compute the gradient of the neural networks [123]. Our AADF model is more robust than neural networks under such attacks because CFS is based on non-differentiable module without performing the gradient-based adjustment. Thus typical gradient-based attacks cannot be used to attack our author attribution model. However, a recent study has shown that it is possible to use adversarial examples to compromise tree-based models [107]. As a result, protecting tree-based author identification models against adversarial examples is another future direction.

7.3.4 Online Learning for Author Identification

One important future issue is performing author identification on data streams for which the environment is open and may change arbitrarily or even adversarially. In the streaming author identification environment, the joint distribution between writing features and the target suspects changes. This problem is referred to as concept drift [108]. The performance of our author identification techniques will drop if we ignore the distribution change, which is not desirable. Online learning can deal with evolving data streams [109]. Therefore, possible solutions are to combine our author identification models with dynamic update methods to cope with various concept drift types.

Adaptive random forest (ARF) [128] is an adaptation of the classical random forest, that can deal with evolving data streams. ARF combines batch algorithm traits with various types of concept drift without complicated optimization. Therefore, one possible solution is to use ARF as the base tree ensemble model in our ensemble of ensembles. Autoencoder-based author verification model can be adapted into an online verification model using stochastic training when stochastic gradient descent is adopted [129]. However, there might be other online learning methods leading to better autoencoder-based author verification results; we leave it for future exploration.

8. Reference

[1] V. Benjamin and H. Chen, “Developing understanding of hacker language through the use of lexical semantics,” in *IEEE International Conference on Intelligence and Security Informatics (ISI)* , 2015.

[2] A. Abbasi and H. Chen. 2008, “Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace,” *ACM Transactions on Information Systems*, 2008.

[3] A. Rocha, J.S. Walter, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. RB Carvalho, and E. Stamatatos, “Authorship attribution for social media forensics,” *IEEE Transactions on Information Forensics and Security*, 2016.

[4] S. Segarra, M. Eisen, and A. Ribeiro, “Authorship attribution through function word adjacency networks,” *IEEE Transactions on Signal Processing*, 2015.

[5] V. Benjamin, B. Zhang, J. F. Nunamaker Jr, and H. Chen, “Examining hacker participation length in cybercriminal Internet-relay-chat communities,” *Journal of Management Information Systems*, 2016.

[6] V. Benjamin and H. Chen, “Time-to-event modeling for predicting hacker IRC community participant trajectory,” in *IEEE International Conference on Intelligence and Security Informatics*, 2016.

[7] S. Mittal, D. K. Prajit Kumar, V. Mulwad, A. Joshi, and T. Finin, “Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities,” in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM),* 2016.

[8] V. Benjamin and H. Chen, “Developing understanding of hacker language through the use of lexical semantics,” in *IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2015.

[9] The Register Security. UK cops: How we sniffed out convicted AnonOps admin 'Nerdo'. https://www.theregister.co.uk/2012/12/14/uk_anon_investigation/, 2012.

[10] B. C. Boatwright, D. L. Linvill, and P. L. Warren, “Troll Factories: The Internet Research Agency and State-Sponsored Agenda Building,” *Resource Centre on Media Freedom in Europe*, 2018.

[11] Y. Xia, J. Lukito, Y. Zhang, C. Wells, S. Kim, and C. Tong, “Disinformation, performed: self-presentation of a Russian IRA account on Twitter,” *Information, Communication & Society*, 2019.

[12] P. Jachim, F. Sharevski, and P. Treebridge, “TrollHunter [Evader]: Automated Detection [Evasion] of Twitter Trolls During the COVID-19 Pandemic,” in *New Security Paradigms Workshop*, 2020.

[13] P. Jachim, F. Sharevski, and E. Pieroni, “TrollHunter2020: Real-time Detection of Trolling Narratives on Twitter During the 2020 US Elections,” in *ACM Workshop on Security and Privacy Analytics*, 2021.

[14] S. Shao, C. Tunc, A. Al-Shawi, and S. Hariri, “One-Class Classification with Deep Autoencoder Neural Networks for Author Verification in Internet Relay Chat,” in *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, 2019.

[15] S. Shao, C. Tunc, A. Al-Shawi, and S. Hariri, “Automated Twitter Author Clustering with Unsupervised Learning for Social Media Forensics,” in *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, 2019.

[16] S. Shao, C. Tunc, A. Al-Shawi, and S. Hariri. “An Ensemble of Ensembles Approach to Author Attribution for Internet Relay Chat Forensics,” *ACM Transactions on Management Information Systems (TMIS)*, 2020

[17] S. Shao, C. Tunc, A. Al-Shawi, and S. Hariri, “Autonomic Author Identification in Internet Relay Chat (IRC),” in *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, 2018

[18] S. Shao, C. Tunc, P. Satam, and S. Hariri, “Real-time irc threat detection framework,” in *IEEE International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, 2017.

[19] J. Bernard, S. Shao, C. Tunc, H. Kheddouci, and S. Hariri, "Quasi-cliques analysis for IRC channel thread detection," in *International Conference on Complex Networks and their Applications*, 2018.

[20] M. H. Altakrori, F. Iqbal, B. Fung, S. H. Ding, and A. Tubaishat, "Arabic Authorship Attribution: An Extensive Study on Twitter Posts," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 2018.

[21] G. Baron, "Comparison of cross-validation and test sets approaches to evaluation of classifiers in authorship attribution domain," in *International Symposium on Computer and Information Sciences*, 2016.

[22] D. Madigan, A. Genkin, D. Lewis, S. Argamon, D. Fradkin, and L. Ye, "Author identification on the large scale," in *Meeting of the Classification Society of North America (CSNA)*, 2005.

[23] K. Luyckx, and W. Daelemans, "Authorship attribution and verification with many authors and limited data," in *International Conference on Computational Linguistics-Volume 1, Association for Computational Linguistics*, 2008.

[24] M. Brocardo, I. Traore, and I. Woungang, "Authorship verification of e-mail and tweet messages applied for continuous authentication," *Journal of Computer and System Sciences*, 2015.

[25] S. Ben-David, "Clustering-what both theoreticians and practitioners are doing wrong," in *AAAI Conference on Artificial Intelligence*, 2018.

[26] B. A. Pimentel, and A. C. Carvalho, "A Meta-learning approach for recommending the number of clusters for clustering algorithms," *Knowledge-Based Systems*, 2020

[27] C. Pasquini, I. Amerini, and G. Boato, "Media forensics on social media platforms: a survey," *EURASIP Journal on Information Security*, 2021.

[28] J. H. Kietzmann, K. Hermkens, I. P. McCarthy, and B. S. Silvestre, "Social media? Get serious! Understanding the functional building blocks of social media," *Business horizons*, 2011.

[29] A. Abbasi and H. Chen, "Applying authorship analysis to extremist-group web forum messages," IEEE Intell. Syst, 2005.

[30] R. Layton, S. McCombie, and P. Watters, “Authorship attribution of IRC messages using inverse author frequency,” in *IEEE Cybercrime and Trustworthy Computing Workshop*, 2012.

[31] V. Benjamin, W. Li, T. Holt, and H. Chen, “Exploring threats and vulnerabilities in hacker web: Forums, IRC and carding shops,” in *IEEE international conference on intelligence and security informatics (ISI)*, 2015.

[32] 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 *International conference on World Wide Web*, 2012.

[33] L. Ilias and I. Roussaki, “Detecting malicious activity in Twitter using deep learning techniques,” *Applied Soft Computing*, 2021.

[34] IBM Watson Personality Insights. https://console.bluemix. net/docs/services/personality-insights, 2018.

[35] IBM Watson Assistant. https://www.ibm.com/cloud/watson-assistant/ , 2018.

[36] R. Zheng, J. Li, H. Chen, and Z. Huang, “A framework for authorship identification of online messages: Writing-style features and classification techniques,” *J. Amer. Soc. Inf. Sci. Technol*, 2006.

[37] R. H. R. Tan and F. S. Tsai, “Authorship identification for online text,” in *International Conference on Cyberworlds (CW’10),* 2010.

[38] T. Solorio, S. Pillay, S. Raghavan, and M. Montes-Gomez, “Modality specific meta features for authorship attribution in web forum posts,” in *International Joint Conference on Natural Language Processing*, 2011.

[39] E. Stamatatos, N. Fakotakis, and G. Kokkinakis, “Automatic text categorization in terms of genre and author,” *Computational linguistics*, 2000.

[40] M. Koppel and J. Schler, “Authorship verification as a one-class classification problem,” in *International conference on Machine learning*, 2004.

[41] M. Koppel, J. Schler, and E. Bonchek-Dokow, “Measuring differentiability: Unmasking pseudonymous authors,” *Journal of Machine Learning Research*, 2007.

[42] H. J. Escalante, M. Montes, and L. Villaseñor, “Particle swarm model selection for authorship verification,” in *Iberoamerican Congress on Pattern Recognition*, 2009.

[43] S. Barbon, R. A. Igawa, and B. B. Zarpelão, “Authorship verification applied to detection of compromised accounts on online social networks,” *Multimedia Tools and Applications*, 2017.

[44] M. Litvak, “Deep Dive into Authorship Verification of Email Messages with Convolutional Neural Network,” in *Annual International Symposium on Information Management and Big Data*, 2018.

[45] B. Boenninghoff, R. M. Nickel, S. Zeiler, and D. Kolossa, “Similarity Learning for Authorship Verification in Social Media,” *in International Conference on Acoustics, Speech and Signal Processing*, 2019.

[46] K. Luyckx, W. Daelemans, and E. Vanhoutte, “Stylogenetics: Clustering-based stylistic analysis of literary corpora,” in *International Conference on Language Resources and Evaluation (LREC)*, 2006.

[47] F. Iqbal, H. Binsalleeh, B. Fung, and M. Debbabi, “Mining writeprints from anonymous e-mails for forensic investigation,” *Digital Investigation*, 2010.

[48] R. Layton, P. Watters, and R. Dazeley, “Automated unsupervised authorship analysis using evidence accumulation clustering,” *Natural Language Engineering*, 2013.

[49] H. Gómez-Adorno, C. Martín-del-Campo-Rodríguez, G. Sidorov, Y. Alemán, D. Vilariño, and D. Pinto. “Author Clustering using Hierarchical Clustering Analysis,” in *International Conference of the Cross-Language Evaluation Forum*, 2017.

[50] G. Inches, M. Harvey, and F. Crestani, “Finding participants in a chat: Authorship attribution for conversational doc- uments,” in *International Conference on Social Computing*, 2013.

[51] R. Schwartz, O. Tsur, A. Rappoport, and M. Koppel, “Authorship attribution of micro-messages,” in *Conference on Empirical Methods in Natural Language Processing*, 2013.

[52] M. Bhargava, P. Mehndiratta, and Krishna Asawa, "Stylometric analysis for authorship attribution on twitter," in *International Conference on Big Data Analytics*, pp. 37-47. Springer, Cham, 2013.

[53] C. Suman, A. Raj, S. Saha, and P. Bhattacharyya, "Authorship Attribution of Microtext Using Capsule Networks," *IEEE Transactions on Computational Social Systems*, 2021.

[54] S. Alterkavı and H. Erbay, "Novel authorship verification model for social media accounts compromised by a human," *Multimedia Tools and Applications*, 2021.

[55] J. Yan, and S. J. Matthews, "Applying clustering algorithms to determine authorship of chinese twitter messages," in *IEEE MIT Undergraduate Research Technology Conference (URTC),* 2016.

[56] O. Zulfiqar, Y. Chang, P. Chen, K. Fu, C. Lu, D. Solnick, and Y. Li, "RISECURE: Metro Incidents And Threat Detection Using Social Media," in IEEE/ACM *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2020.

[57] M. Sauter, "The Coming Swarm: DDOS Actions, Hacktivism, and Civil Disobedience on the Internet," *Bloomsbury Academic*, 2014.

[58] Twitter API. https://developer.twitter.com/en/docs/twitter-api , 2020.

[59] Twint. https://github.com/twintproject/twint, 2020.

[60] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Conference on Empirical Methods in Natural Language Processing*, 2013.

[61] W. Wu, J. Zhou, Y. Xiang, and L. Xu. 2013, "How to achieve non-repudiation of origin with privacy protection in cloud computing," *J. Comput. Syst. Sci*, 2013.

[62] S. Samtani and H. Chen, "Using social network analysis to identify key hackers for keylogging tools in hacker forums," in *IEEE Conference on Intelligence and Security Informatics (ISI)*, 2016.

[63] S. Samtani, R. Chinn, H. Chen, and J. F. Nunamaker Jr, "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence," *J. Mana. Inf. Syst*, 2017.

[64] S. Samtani, K. Chinn, C. Larson, and H. Chen, "AZSecure hacker assets portal: Cyber threat intelligence and malware analysis," in *IEEE Conference on Intelligence and Security Informatics (ISI),* 2016.

[65] F. Amuchi, A. Al-Nemrat, M. Alazab, and R. Layton, "Identifying cyber predators through forensic authorship analysis of chat logs," in *IEEE Cybercrime and Trustworthy Computing Workshop*, 2012.

[66] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for information Science and Technology*, 2009

[67] A. Białecki, R. Muir, G. Ingersoll, and L. Imagination, "Apache lucene 4," in *SIGIR Workshop on Open Source Information Retrieval*, 2017

[68] A. Rajaraman and J. D. Ullman, "Mining of Massive Datasets," Cambridge University Press, 2011.

[69] R. Sedgewick and K. Wayne, "Algorithms fourth edition," https://algs4.cs.princeton.edu/35applications/stopwords.txt.

[70] NIST. Glossary of key information security terms. https://csrc.nist.gov/glossary. 2018.

[71] IRC Abbreviation. Introduction to IRC Abbreviations. http://w w w.ircbeginner.com/ircinfo/ abbreviations.html, 2018.

[72] IRC Emoticons. Introduction to IRC Emoticons. http://w w w.ircbeginner.com/ircinfo/emoticons.html .

[73] D. A. Cobb-Clark and S. Schurer, "The stability of big-five personality traits," *Econ. Lett*, 2012.

[74] V. Benjamin, and H. Chen, "Identifying language groups within multilingual cybercriminal forums," in *IEEE Conference on Intelligence and Security Informatics (ISI),* 2016.

[75] Z. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in International Joint Conference on Artificial Intelligence , 2017.

[76] L. Breiman, "Random forests", *Machine Learning*, 2001.

[77] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine. Learning*, 2006.

[78] K. Luyckx and W. Daelemans, "Authorship attribution and verification with many authors and limited data," in *International Conference on Computational Linguistics*, 2008.

[79] G. Hirst and O. Feiguina, "Bigrams of syntactic labels for authorship discrimination of short texts," *Literary and Linguistic Computing*, 2007.

[80] X. Liu, J. Wu, and Z. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B ,* 2009.

[81] Imbalanced-learn. https://imbalanced-learn.org/stable/, 2020.

[82] P. Y. Du and N. Zhang, Hacker Web Forum Collection: IRC conversation for channel. University of Arizona Artificial Intelligence Lab, AZSecure-data, Director Hsinchun Chen. http://www.azsecure-data.org/, 2018.

[83] Scikit-Learn API. https://scikit-learn.org/stable/, 2020.

[84] C.D. Manning, P. Raghavan and H. Schuetze, "Introduction to Information Retrieval," *Cambridge University Press*, 2008.

[85] C-C. Chang and C-J. Lin. 2011, "LIBSVM: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST),* 2011.

[86] Z. Zhou, "Ensemble Methods: Foundations and Algorithms," *CRC, Boca Raton, FL*, 2012.

[87] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, ..., T. Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017.

[88] G. Louppe, "Understanding Random Forests: From Theory to Practice," *Dissertation. University of Liege, Belgium*, 2014.

[89] F. Zheng and I. Geoffrey, "A comparative study of semi-naive bayes methods in classification learning," in *Australasian Data Mining Workshop*, 2005.

[90] S. Khan, and M. G. Madden, "A survey of recent trends in one class classification," in *Irish conference on artificial intelligence and cognitive science*, 2009.

[91] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*, 2018.

[92] C. Zhou, and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[93] D. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint* ,2014.

[94] A. Sharma and U. Ghose, "Sentimental analysis of twitter data with respect to general elections in india," *Procedia Computer Science*, 2020.

[95] SocialMediaToday, https://www.socialmediatoday.com/social-networks/sarah-snow/2015-07-08/get-out-your-twittonary-twitter-abbreviations-you-must-know, 2015.

[96] Emojitracker: realtime emoji use on twitter, http://emojitracker.com/, 2019.

[97] S.K. Ng, T. Krishnan, and G. J. McLachlan, "The EM algorithm," in *Handbook of computational statistics*, 2012.

[98] F. Eibe, M. A. Hall, and I. H. Witten, "The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques," *Morgan Kaufmann*, 2016.

[99] D. Pelleg and A. W. Moore, "X-means: extending k-means with efficient estimation of the number of clusters," in *International conference on Machine learning* , 2000.

[100] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. "Estimating the support of a high-dimensional distribution," *Neural computation*, 2001

[101] F. Liu, K. Ting, and Z. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2012.

[102] D. Estival, T. Gaustad, S. B. Pham, W. Radford, and B. Hutchinson, "Author profiling for English emails," in *Conference of the Pacific Association for Computational Linguistics*, 2007.

[103] K. Kavuri and M. Kavitha, “A Stylistic Features Based Approach for Author Profiling,” in *Recent Trends in Communication and Intelligent Systems*, 2020.

[104] M. Brennan, S. Afroz, and R. Greenstadt, “Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity,” *ACM Transactions on Information and System Security (TISSEC)*, 2012.

[105] A. McDonald, S. Afroz, A. Caliskan, A. Stolerman, and R. Greenstadt, “Use fewer instances of the letter “i”: Toward writing style anonymization,” in *International Symposium on Privacy Enhancing Technologies Symposium*, 2012.

[106] D. Castro-Castro, R. O. Bueno, and R. Muñoz, “Author Masking by Sentence Transformation,” in *International Conference of the Cross-Language Evaluation Forum,* 2017.

[107] H. Chen, H. Zhang, D. Boning, and C-J. Hsieh, “Robust Decision Trees Against Adversarial Examples,” in *International Conference on Machine Learning*, 2019

[108] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, “Learning in nonstationary environments: A survey,” *IEEE Computational Intelligence Magazine*, 2015.

[109] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and Abdelhamid Bouchachia, “A survey on concept drift adaptation,” *ACM computing surveys*, 2014.

[110] K. Sun, S. Huang, D. Wong, and S. Jang, “Design and application of a variable selection method for multilayer perceptron neural network with LASSO,” *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[111] L. Jiang, S. Wang, Chaoqun Li, and Lungan Zhang. "Structure extended multinomial naive Bayes." Information Sciences 329 (2016): 346-356.

[112] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” *ACM SIGMOD International Conference on Management of Data*, 2000.

[113] J. Tang, Z, Chen, A.WC. Fu, and D.W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, 2002.

[114] H. Fan, O. R. Zaïane, A. Foss, and J. Wu, "A nonparametric outlier detection for effectively discovering top-n outliers from engineering data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2006.

[115] C. Wu, S. Shao, C. Tunc, and S. Hariri, "Video anomaly detection using pre-trained deep convolutional neural nets and context mining," in *International Conference on Computer Systems and Applications (AICCSA),* 2020.

[116] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, 2015.

[117] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, 2003.

[118] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2012.

[119] D. A. Reynolds, Douglas, "Gaussian mixture models," *Encyclopedia of Biometrics*, 2009.

[120] C. Leistner, A. Saffari, and H. Bischof, "MIForests: Multiple-instance learning with randomized trees," in *European Conference on Computer Vision*, 2010.

[121] M. Sabzevari, G. Martínez-Muñoz, and A. Suárez, "A two-stage ensemble method for the detection of class-label noise," *Neurocomputing*, 2018.

[122] P. Juola and D. Vescovi, "Analyzing stylometric approaches to author obfuscation," in *IFIP International Conference on Digital Forensics*, 2011.

[123] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[124] X. Gao, J. Szep, P. Satam, S. Hariri, S. Ram, and J. J. Rodriguez, "Spatio-Temporal Processing for Automatic Vehicle Detection in Wide-Area Aerial Video," IEEE Access, 2020.

[125] H. Liu and G. Ditzler, "A semi-parallel framework for greedy information-theoretic feature selection," *Information Sciences*, 2019.

[126] H. Liu and G. Ditzler, "Data poisoning against information-theoretic feature selection," *Information Sciences*, 2021

[127] C. Zuo, Y. Zhao, and R. Banerjee, "Style Change Detection with Feed-forward Neural Networks," in *Conference and Labs of the Evaluation Forum*, 2019.

[128] H. M. Gomes, A. Bifet, J. Read, J. Barddal, F. Enembreck, B. Pfharinger, H. Geoff, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, 2017.

[129] Q. Cui, Z. Gong, W. Ni, Y. Hou, X. Chen, X. Tao, and P. Zhang, "Stochastic online learning for mobile edge computing: Learning from changes," *IEEE Communications Magazine*, 2019.

[130] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *SIAM international conference on data mining*, 2017.

[131] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *International Conference on Machine learning*, 2006.

[132] J. Foulds and E. Frank, "A review of multi-instance learning assumptions," *The knowledge Engineering Review*, 2010.

[133] M. Wang, H. Yu, and F. Min, "Noise label learning through label confidence statistical inference," *Knowledge-Based Systems*, 2021.