DISSERTATION

AUTOMATING INVESTIGATIVE PATTERN DETECTION USING MACHINE LEARNING & GRAPH PATTERN MATCHING TECHNIQUES

Submitted by

Shashika R. Muramudalige

Department of Electrical & Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2022

Doctoral Committee:

Advisor: Anura P. Jayasumana

Indrakshi Ray
Ryan G. Kim
Haonan Wang

ABSTRACT

AUTOMATING INVESTIGATIVE PATTERN DETECTION USING MACHINE LEARNING & GRAPH PATTERN MATCHING TECHNIQUES

Identification and analysis of latent and emergent behavioral patterns are core tasks in investigative domains such as homeland security, counterterrorism, and crime prevention. Development of behavioral trajectory models associated with radicalization and tracking individuals and groups based on such trajectories are critical for law enforcement investigations, but these are hampered by sheer volume and nature of data that need to be mined and processed. Dynamic and complex behaviors of extremists and extremist groups, missing or incomplete information, and lack of intelligent tools further obstruct counterterrorism efforts. Our research is aimed at developing state-of-the-art computational tools while building on recent advances in machine learning, natural language processing (NLP), and graph databases.

In this work, we address the challenges of investigative pattern detection by developing algorithms, tools, and techniques primarily aimed at behavioral pattern tracking and identification for domestic radicalization. The methods developed are integrated in a framework, Investigative Pattern Detection Framework for Counterterrorism (INSPECT). INSPECT includes components for extracting information using NLP techniques, information networks to store in appropriate databases while enabling investigative graph searches, and data synthesis via generative adversarial techniques to overcome limitations due to incomplete and sparse data. These components enable streamlining investigative pattern detection while accommodating various use cases and datasets. While our outcomes are beneficial for law enforcement and counterterrorism applications to counteract the threat of violent extremism, as the results presented demonstrate, the proposed framework is adaptable to diverse behavioral pattern analysis domains such as consumer analytics, cybersecurity, and behavioral health.

Information on radicalization activity and participant profiles of interest to investigative tasks are mostly found in disparate text sources. We integrate NLP approaches such as named entity recognition (NER), coreference resolution, and multi-label text classification to extract structured information regarding behavioral indicators, temporal details, and other metadata. We further use multiple text pre-processing approaches to improve the accuracy of data extraction. Our training text datasets are intrinsically smaller and label-wise imbalanced, which hinders direct application of NLP techniques for better results. We use a transfer learning-based, pre-trained NLP model by integrating our specific datasets and achieve noteworthy improvement in information extraction.

The extracted information from text sources represents a rich knowledge network of populations with various types of connections that needs to be stored, updated, and repeatedly inspected for emergence of patterns in the long term. Therefore, we utilize graph databases as the foremost storage option while maintaining the reliability and scalability of behavioral data processing. To query suspicious and vulnerable individuals or groups, we implement investigative graph search algorithms as custom stored procedures on top of graph databases while verifying the ability to operate at scale. We use datasets in different contexts to demonstrate the wide-range applicability and the enhanced effectiveness of observing suspicious or latent trends using our investigative graph searches.

Investigative data by nature is incomplete and sparse, and the number of cases that may be used for training investigators or machine learning algorithms is small. This is an inherent concern in investigative and many other contexts where the data collection is tedious, available data is limited and also may be subjected to privacy concerns. Having large datasets is beneficial to social scientists and investigative authorities to enhance their skills, and to achieve more accuracy and reliability. A not so small training data volume is also essential for application of the latest machine learning techniques for improved classification and detection. In this work, we propose a generative adversarial network (GAN) based approach with novel feature mapping techniques to synthesize additional data from a small and sparse data set while preserving the statistical characteristics. We also compare our proposed method with two likelihood approaches. i.e., multi-variate

Gaussian and regular-vine copulas. We verify the robustness of the proposed technique via a simulation and real-world datasets representing diverse domains.

The proposed GAN-based data generation approach is applicable to other domains as demonstrated with two applications. Initially, we extend our data generation approach by contributing to a computer security application resulting in improved phishing websites detection with synthesized datasets. We merge measured datasets with synthesized samples and re-train models to improve the performance of classification models and mitigate vulnerability against adversarial samples. The second was related to a video traffic classification application in which to the data sets are enhanced while preserving statistical similarity between the actual and synthesized datasets. For the video traffic data generation, we modified our data generation technique to capture the temporal patterns in time series data. In this application, we integrate a Wasserstein GAN (WGAN) by using different snapshots of the same video signal with feature-mapping techniques. A trace splitting algorithm is presented for training data of video traces that exhibit higher data throughput with high bursts at the beginning of the video session compared to the rest of the session. With synthesized data, we obtain 5 - 15/% accuracy improvement for classification compared to only having actual traces.

The INSPECT framework is validated primarily by mining detailed forensic biographies of known jihadists, which are extensively used by social/political scientists. Additionally, each component in the framework is extensively validated with a Human-In-The-Loop (HITL) process, which improves the reliability and accuracy of machine learning models, investigative graph algorithms, and other computing tools based on feedback from social scientists. The entire framework is embedded in a modular architecture where the analytical components are implemented independently and adjustable for different requirements and datasets. We verified the proposed framework's reliability, scalability, and generalizability with datasets in different domains. This research also makes a significant contribution to discrete and sparse data generation in diverse application domains with novel generative adversarial data synthesizing techniques.

ACKNOWLEDGEMENTS

There are many people behind this journey. First, I am extremely grateful to my supervisor Prof. Anura Jayasumana for selecting me as his student, for his invaluable advice, continuous support, and patience during this journey. I also want to thank him for giving me the opportunity to work with his colleagues and contribute to diverse research problems. I would like to thank my committee members Prof. Indrakshi Ray, Prof. Haonan Wang, and Prof. Kim Ryan for their encouragement and for the opportunity given to collaborate with them while sharing their immense knowledge and plentiful experience. I am also forever grateful to Dr. Benjamin W.K. Hung for his support, encouragement, and spending hours reviewing my work within his busy schedules.

I would also like to thank Prof. Jytte Klausen and her team for their continuous support in reviewing my work and sharing their invaluable social science expertise to succeed in my dissertation. Further, I would like to express my gratitude to Dr. Hossein Shirazi for allowing me to collaborate with his research, for sharing his knowledge, and for reviewing my work. I am thankful to Dr. Kanchana Thilakarathna, Dr. Guillaume Jourjon, and Mr. Chamara Madarasingha for letting me contribute to their research work, reviewing my work, and sharing their immense knowledge.

I am grateful to the Electrical and Computer Engineering Department, Colorado State University for granting me a fellowship and two scholarships during the degree program. I would like to thank my friends and lab mates for the cherished time spent together and for sharing their experience and knowledge. I am also forever grateful to my Master's degree supervisor Dr. Dilum Bandara for his invaluable advice, constructive criticisms, and spending hours reviewing my work.

Finally yet importantly, I am forever grateful to my parents and my love Sachintha Mendis for their love and support throughout this journey. I am thankful to my sister and brother-in-law for their love and support, and for taking care of my parents throughout these years. I am also forever grateful to all the teachers and mentors I met in my college and the university.

This work was supported by the U.S. Department of Justice, Office of Justice Programs/National Institute of Justice under Award 2017-ZA-CX-0002. Opinions or points of view expressed in this article are those of the authors and do not necessarily reflect the official position of policies of the U.S. Department of Justice.

DEDICATION

I would like to dedicate this thesis to all people in the beautiful island Sri Lanka who pay taxes to facilitate free education for all students.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Investigative pattern analysis and detection generally refers to recognizing patterns and regularities of communication and behavioral activities. It facilitates forensics as well as surveillance of suspicious activities and trends. Investigative pattern analysis is applicable in diverse domains and is still an emerging area with many challenges related to dynamic, incomplete, and unreliable data, and need for handling diverse types of information. The growth of social networking has resulted in the generation massive amounts of data applicable in investigative pattern detection. Large-scale predictive and analytical behavioral models can now be conceived due to the advancements of machine learning and big data concepts in recent years. Identifying and analyzing latent and emergent behavioral patterns are noteworthy in homeland security, consumer analytics, cybersecurity, behavioral health, and other domains where patterns of behavioral indicators provides expressive insights. Such data can be represented as dynamic graphs (Carley, Pfeffer, Morstatter, & Liu, 2014; Klausen, Libretti, Hung, & Jayasumana, 2018), and are often observable in interactions via social networks (Klausen, Marks, & Zaman, 2018; Lara-Cabrera et al., 2017). Organizations and law enforcement bodies continually seek to detect insider threats using technical indicators and alerts recorded over time (CERT Division Software Engineering Institute, 2019). Risk assessment protocols in cybersecurity require techniques to observe sequences of suspicious activities within considerable time frames. Businesses track an individual's online purchase habits to determine the potential for future purchases (Edelman & Singer, 2019, 2015). Identification of behaviors that precede suicide is of vital interest (Jashinsky et al., 2014; Olson, 2011) to mental health professionals. Graph-based models that allow mining and tracking of such behavior appear to be a promising approach. In this research, we are particularly interested in identifying homegrown violent extremism, which is a cause of concern in many countries (Klausen, Campion, Needle, Nguyen, & Libretti, 2016).

The statistics about violent extremist attacks indicate the severity of the threat by violent extremists in the USA (Valverde, 2017) and Europe (Nesser, Stenersen, & Oftedal, 2017). In 2015-2017, there was a significant rise of the violent extremist attacks in the western world, which increased the terrorist activities in many countries. Previously, the 9/11 attacks resulted in the largest number of deaths, about 3,000 people were killed in the United States caused by violent extremism (Valverde, 2017). Since then, from Sept. 12, 2001, to Dec. 31, 2016, there were 85 attacks in the USA by violent extremists resulting in 225 deaths based on the statistics from the U.S. Extremist Crime Database (Freilich, Chermak, Gruenewald, & Parkin, 2014). Terrorism experts reveal the threat of jihadist attacks against the US and the West persist due to the terrorist organization's continued ability to attract followers and inspire or direct attacks (Hoffman, n.d.; Homeland Security Committee, 2018; Muramudalige, Hung, Jayasumana, & Ray, 2019). The US House of Representatives Homeland Security Committee reported in October 2018 a 63% increase in the number of ISIS-inspired attacks within the last few years, just as the so-called Caliphate was dwindling in size (Homeland Security Committee, 2018). In 2019, Easter Sunday attacks in Sri Lanka by homegrown violent extremists killed more than 250 people (Kapur, 2019), which implies the continuous danger of violent extremism has been spread not only in the western countries but all over the world. Therefore, identifying violent extremists and their trajectories in advance in a timely manner is important to overcome these attacks and make the world a safer place.

Radicalization is a process of having extreme political, social, or religious ideals that could be enforced to violent extremism. The term "radicalization" is commonly used but controversial. It is shorthand for "radicalization to violent extremism," which implies a process view of how individuals move from beliefs to actions (Borum, 2012). For this reason, a recent guide issued by the Office of the Director of National Intelligence uses the term "mobilization" to indicate the overt behavioral dynamics associated with growing radicalization leading to terrorism-related actions (The office of director of national intelligence, 2019). A person gets radicalized through a sequence of activities; these include the consumption of extremist ideas and propaganda through internet sources and the association with other radicalized peer groups (King & Taylor, 2011; Klausen et

al., 2016; Klausen, Libretti, et al., 2018). Detecting homegrown violent extremists is a tedious and challenging task because they live inside the ordinary society and may not be exposed until the day of the attack. Social and political scientists, and law enforcement authorities seek to identify homegrown violent extremists using specific behavioral indicators that in some cases only slightly deviate from normal people's behavior. A recent booklet issued by the Office of the Director of National Intelligence (The office of director of national intelligence, 2019) with collaboration by the FBI, the US National Counterterrorism Center, and the US Department of Homeland Security, describes the lists of observable indicators of potential violent extremists. It also defines a homegrown violent extremist (HVE) as a person who advocates, engages in, or is preparing to engage in or support terrorist activities in furtherance of a foreign terrorist organization's objectives, but who is acting independently of foreign terrorist direction (The office of director of national intelligence, 2019). The complicated behavioral patterns of homegrown extremists make the investigations extremely challenging. The behavioral indicators are sometimes individually innocuous and it requires further investigation and surveillance to detect links among other persons to identify the violent extremist activities and their networks.

1.1 Motivation

Counterterrorism and law enforcement professionals have been encountering diverse challenges in trying to detect violent extremism. One of the problems is not too little but too much information. In November 2019, Russell E. Travers, the then acting director of the National Counterterrorism Center, described the problem in a speech he made at the Washington Institute for Near East Policy. "I've spent my entire career working analytic issues and will say unequivocally that counterterrorism has the worst signal-to-noise ratio of any discipline I've ever worked," he said. He went on to specify "my ops [operations] center receives something in excess of 10,000 terrorism-related intelligence reports a day through which we need to sift. And those 10,000 reports contain 16,000 names. Daily." (R. E. Travers, National Counterterrorism Center, 2019). Investigators cannot process manually heavy volumes of data on the scale described in the above quote.

Furthermore, many of the erroneous queries lead from focusing on accurate plots that may cause serious troubles.

Another significant challenge is that initial tasks and/or attacks have been conducted as groups in many cases. Therefore, finding such attacks in advance is an intimidating challenge to detect how indicators are emerging among individuals or groups in a large population. An FBI study in 2018 from 63 cases of "active shooter" found no demographic factors that were sufficient to detect in advance the individuals posing a threat of extreme violence (Silver, Simons, & Craun, 2018). However, the FBI explored that in nearly all the cases a bystander, observed *radicalized behaviors* and sometimes *preparatory planning* but did not report their observations. Three people on average observed such behaviors, and the perpetrators displayed 4 or 5 concerning behaviors, including discussing their discretions. In 75% of the cases, the concerning behaviors were initially monitored between six months to two or more years before the incident/attack. Obviously, identifying behavioral indicators is important in preventing terrorist attacks in advance, and scalable and efficient tools are required for law enforcement to track the radicalized individuals with sufficient lead time to prevent, interfere or intercept. These dynamic behavioral indicators deliver more insights in a temporal context and relationships among indicators/individuals facilitate rich information graph models. Such dynamic graph models provide a powerful tool to represent such diverse information and facilitate these types of analysis.

Further, these dynamic approaches are required to evaluate frequently with expertized knowledge from relevant bodies because especially the concept of "risk" is challenging to apply or quantify. Threat prevention needs immediate decisions in response to limited information in an operational and dynamic environment. These assessments benefit from probabilistic modeling based on evidence to distinguish the regular behaviors and actions associated with the type of violent extremism that raised concerns. The purpose of a dynamic approach to threat assessments is instead to evaluate any changes in an individual's behavior that suggest an increased concern to commit violent extremist activity (Freeland, Klausen, & Pagé, 2019).

The following motivating example, described in (B. W. Hung, Jayasumana, & Bandara, 2019), demonstrates why graph-based, dynamic analysis methods are needed to support counterterrorism efforts. The behavioral indicators and association graph of the extremists, involved in the San Bernardino terrorist attack in 2015 are shown in Figure 1.1 (B. W. Hung, Jayasumana, & Bandara, 2018). The perpetrators of the attack, Syed Farook and Tashfeen Malik, initially met on an extremist dating site. The two married in Saudi Arabia, then traveled to the U.S. together and subsequently got legally married in Riverside, California. Both had direct contacts with radicalized groups via social media, and both went along to a shooting range for weapons training before the attack. Enrique Marquez, a friend and a former neighbor of Farook, provided the weapons used in the attack. Investigators believe the plot was not detected before the attack because the couple radicalized separately before meeting each other (Times, 2015). Had these individuals been identified as a group, their collective behavior could have signified a sudden extremist development or a threat of an attack. To identify such plots, it is vital to detect associations and collective behaviors. While such efforts are tedious, semi-automated analytics could more efficiently identify and measure groups of people and their shared behavior. With the use of natural language processing techniques, it is becoming possible to extract large knowledge networks (Hung et al., 2019), mining of which require techniques that are scalable and adaptive.

Identifying and distinguishing criminal activities of individuals, gangs and other crime organizations from networks and collections of information, and tracking their latent and emergent behaviors are overwhelming tasks for law-enforcement authorities. The FBI says there are over 30,000 violent street gangs, motorcycle gangs, and prison gangs that are criminally active in the U.S. (Federal Bureau of Investigation (FBI), 2020b). In the recent years, gang violence has been further aggravated by taunting between gang members on social media in the U.S (Blevins et al., 2016). Detecting such activities is non-trivial as individual behaviors may not follow all the profile components or steps, while the group/gang as a whole does. Thus, efficient mechanisms are needed to track partially matching profiles of individuals which taken together satisfy the profile of interest.

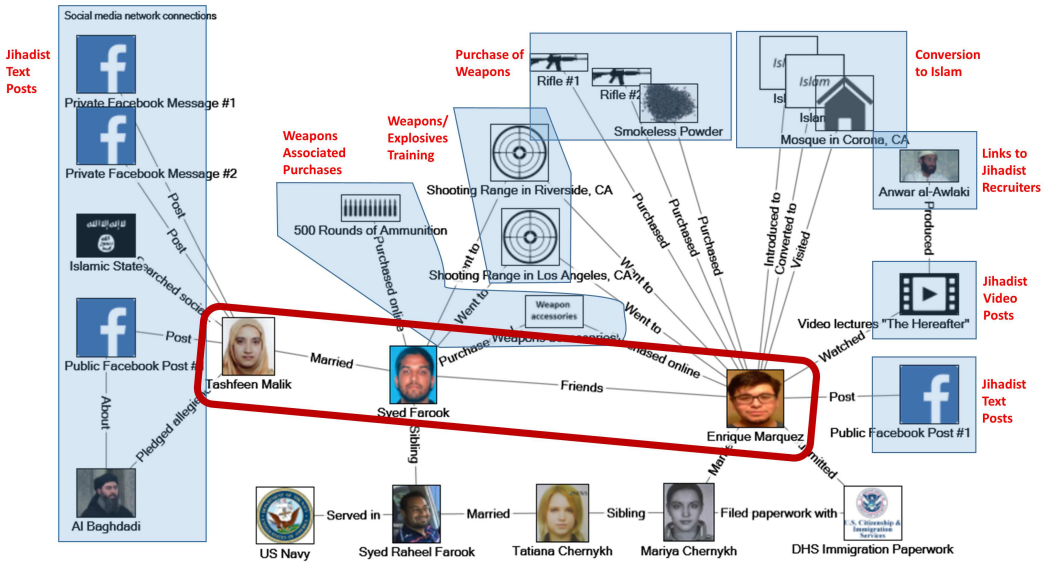


Figure 1.1: San Bernardino Terrorist Attack, 2015. Behavioral indicators and association graph of Syed Farook, Tashfeen Malik, and Enrique Marquez show signals of their collective radicalization and preparations for the attack

These problems motivate us to implement an end-to-end investigative pattern detection framework that consists of various computing tools for automating the process of investigative pattern detection. The social and political experts investigate the behavior of extremists and criminals through their past activities. Investigative data analysis for prediction and prevention often involves similar graph mining operations. Investigative data such as radicalization and criminal data are highly connected, and often contain social media connections among individuals, and also association with suspicious activities, events, and locations. Such data implicitly represents a complex multi-dimensional network that facilitates analysis through complex graph search and pattern matching operations. Some online social network platforms also offer graph search on their networks (Facebook, 2020; Twitter, 2020); however, they allow the use of very limited amounts of data due to scalability, confidentiality, or legal constraints. Moreover, data integrity is a concern in online social networks due to the proliferation of fake accounts. The biographical data used in this study has been manually collected from court documents and other public sources on Ameri-

can homegrown Salafi-jihadist terrorism offenders (Klausen et al., 2016; Klausen, Libretti, et al., 2018).

1.2 Contributions

Administrative bodies and law-enforcement authorities are continuously looking for efficient and robust solutions to detect violent extremists and groups in advance to overcome these attacks as they remain vigilant against future terrorist attacks by investigating and intercepting those on suspected radicalization pathways to violent extremism (Hung et al., 2019). Social and political scientists, law enforcement authorities study existing profiles and already committed attacks to identify the features of behavioral indicators of extremists and model behavioral profiles to determine forthcoming interested individuals and groups. The process of transforming to an extremist is dynamic and complicated because it depends on a wide range of behavioral indicators, personalities, and situations. In our research, we utilize the datasets of Klausen's Western Jihadism Database (WJDB) (Klausen, Libretti, Renugopalakrishnan, & et al, 2020), a collection that includes information on approximately 6,600 individual jihadists of Western origin or residence who have engaged in criminal terrorist action. All the data derives from public sources ranging from court records, government press releases, and autobiographical statements made by the terrorism offenders themselves on social media or in jihadist forums.

In this research, we propose an end-to-end investigative pattern detection framework consisting of several components in automating investigative pattern detection through machine learning and graph pattern matching techniques. The proposed techniques and tools are initially validated with WJDB's data, and the Human-in-the-loop (HITL) plays a significant role in validating the proposed framework where social scientists iteratively provide feedback to improve the accuracy and efficacy of the computational complements (Hung et al., 2019). Further, we implement the framework with a modular architecture where the components are easily pluggable in other investigative detection contexts with different datasets. Our contribution is as follows:

- We propose an end-to-end investigative pattern detection framework that integrates different functional components to automate investigative pattern detection through machine learning and graph pattern matching techniques. The implemented tools are interdependent each other to achieve the process of automating the investigative pattern detection. We developed tools and techniques to extract relevant information in disparate text sources, then the extracted data was modeled as an information network and stored in an appropriate database. We implemented graph search algorithms on top of the data storage to query suspicious individuals and groups. Further, we developed a novel synthetic data generation scheme for small and sparse datasets to apply machine learning and classification techniques. We discuss each component descriptively in the following.

- As investigative information is available mainly in disparate text sources such as court orders, analysis reports, and news articles, information extraction represents a crucial step in this process. We utilized NLP techniques such as named entity recognition, coreference resolution, multi-label text classification to extract radicalization behavioral indicators and other relavent details. We also applied different data pre-processing and cross-validation techniques to address the data-specific challenges in investigative text datasets.

- The extracted data from text sources were arranged as a comprehensive information network consisting of records/behavioral details of extremists and suspicious individuals. Having appropriate data storage is important to store information in long term and query data efficiently. We utilized a graph database, a convenient data store to address the efficiency and scalability of dynamic, large, and heterogeneous knowledge networks.

- We implement investigative graph search to identify suspicious individuals and groups from large information and knowledge networks. The proposed investigative graph search algorithms are built on top of graph databases while addressing the efficiency and scalability as applied to networks. We apply the developed investigative graph search routines over several networks for different investigative domains.

- In investigative and many other contexts, smaller and incomplete datasets mitigate the possibilities of applying machine learning and classification techniques. To address these challenges, we introduce a data generation technique with novel feature mapping techniques for sparse and incomplete data. The proposed architecture is composed of an adversarial autoencoder (AAE) and novel feature mapping techniques.

  We initially applied the proposed data generation to enhance the WJDB (Klausen et al., 2020) extremist profiles, which assist social scientists in extensively studying different behavioral patterns and combinations. We formalize our data generation strategy to generate complex data objects with a hierarchical structure with a smaller set of actual data. We further compare our technique with traditional statistical data generation approaches.

- The synthetic data generation scheme that we generated is also applicable to other domains that lack sufficient data to train machine learning and classification models. We apply and demonstrate our data generation technique in two other applications in different domains where experiences lack data to train machine learning classification models; phishing websites detection and video traces classification. In Phishing website detection, we use an Adversarial Autoencoder (AAE) to generate samples that mimic the phishing websites and provide metrics to assess the quality of the generated samples. We test these samples against models trained with real-world data. Some of the generated samples are able to evade the existing detection model. We then use a portion of these samples in training. The new machine learning models are more robust and have higher accuracy. In other words, real-world phishing site data augmented with AAE synthesized data used for training the model is more effective for phishing detection.

- In video traces classification, we expand our data generation technique to synthesize time-series data. Unlike conventional internet applications such as web browsing and peer-to-peer(P2P), video streaming has dominated the global network traffic for the past few years, raising many challenges for network providers. With the demand for interactive videos, a.k.a

360° videos, resource requirement for video streaming has been further enhanced. Prior identification of these video traffic is helpful for effective provisioning of network resources, but the data encryption and privacy concerns limit such evaluations. Therefore, collecting a significant amount of data to train machine learning classifiers has become a challenge. Hence, we propose a novel Generative Adversarial Network (GAN) based data generation solution to synthesize video streaming data that helps to improve the classification accuracy significantly. Here, we propose a Wasserstein GAN-based approach with appropriate data mapping pipeline for time-series data.

1.3 Outline

The rest of this dissertation is structured as follows. Chapter 2 reviews the related work on tools and techniques implemented in the proposed investigative framework, and applications used for validate the framework. Chapter 3 widely discusses the problem statement and the research objectives. Chapter 4 presents the overall proposed framework; INSPECT: Investigative Pattern Detection Framework for Counterterrorism. Chapter 5 and 6 comprehensively discuss the data extraction using various NLP techniques and proposed investigative graph search, respectively. Chapter 7 generally discusses the novel data generation technique for complex object generation. Chapter 8, 9, and 10 presents the different applications of the proposed data generation method. Chapter 8 describes the radicalization trajectory generation for extremists. Chapter 9 and 10 presents the details of phishing data generation and video traces generation, respectively. At last, Chapter 11 concludes the summary of the research and discusses the future directions.

Chapter 2

Background & Related Work

We comprehensively discuss literature that is related to components implemented and integrated into the investigative pattern detection framework. Therefore, we present the related work on information extraction from disparate text sources, pattern detection, and graph search in heterogeneous social networks. Furthermore, we review the literature in data generation using both statistical and deep learning approaches and data generation applications in different domains that we apply during this work. Initially, we explain the details of different investigative datasets that we mainly utilized in the research.

2.1 Investigative datasets

We mainly used Klausen's Western Jihadism Database (WJDB) [22], a collection that includes information on approximately 6,600 individual jihadists of Western origin or residence who have engaged in criminal terrorist action. All the data originates from public sources ranging from court records, government press releases, and autobiographical statements made by the terrorism offenders themselves on social media or in jihadist forums. Coders/Analysts were trained to read a variety of publicly accessible documentation for evidence of the over 24 distinct behavioral indicators theorized to be associated with radicalization and instructed to record the dates at which such behaviors were publicly observed. This analysis enabled the retrospective estimation of timelines for the radicalization trajectories. The coders manually extracted a core set of sentences and sentence fragments used to create a labeled dataset to implement computing tools in the INSPECT framework. The dataset utilized to train different NLP techniques for information extraction in these text sources. The efforts aided coders in improving their annotation process while validating the labels via NLP models. Further, we converted the WJDB to a graph database (Section 6.9), initially stored in a SQL database, and verified the efficacy of querying graph databases for such highly connected data. Then, we use the same radicalization trajectories in WJDB to initially

implement the data generation technique for small and sparse datasets (Chapter 8). To validate implemented computing tools of INSPECT in other domains, we used different sets of data in various applications, and those were discussed descriptively in later Chapters.

2.2 Information extraction from text sources and natural language processing techniques

Natural language processing and machine learning algorithms in information extraction tasks have been used successfully in other domains such as understanding patient medical profiles in free-text clinical notes (Friedman, Rindflesch, & Corn, 2013; Kreimeyer et al., 2017; Amazon Web Services (AWS), 2021) and detecting cyberbullying (Xu, Jun, Zhu, & Bellmore, 2012; X. Zhang et al., 2016). In the counterterrorism domain, nascent applications include detecting terrorist intentions (Brynielsson et al., 2013) or determining a social media account's state of radicalization (Lara-Cabrera et al., 2017). To our knowledge, there has not been heretofore an effort to classify text for the presence of distinct radicalization indicators in a manner that is consistent with a risk assessment protocol developed by terrorism experts (Klausen, 2016; Klausen, Libretti, et al., 2018).

Moreover, we note that this information extraction effort is intended to support a growing body of work to develop a capability that assists analysts in rapidly mining law enforcement and intelligence databases for cues and risk indicators as well as dynamically assessing individualized violent extremism risk at scale through computational modeling (B. Hung, Jayasumana, & Bandara, 2017; B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019; Muramudalige et al., 2019). The underlying approach leverages advances in graph pattern matching over a heterogeneous knowledge graph in order to identify those on a trajectory of extremist violence according to a risk assessment protocol. Our work here supports the construction of such knowledge graphs by extracting structured information from law enforcement and intelligence reports.

Our effort to use computational modeling to assist analysts and case workers responsible for sorting diverse pools of people thought to present a risk to public safety is inspired in part by

comparable efforts in public health management approaches to preventive intervention. Notable examples in this line of investigation is the use of NLP techniques for identifying high risk child abuse cases (Castellani, Griffiths, Rajaram, & Gunn, 2018).

The research involved in fulfilling these assumptions is often underestimated. The use of digital technology and NLP techniques for risk assessment rests on two premises: 1) the risk factors and overt behaviors associated with the specific pathology have to be known, and 2) the ability of algorithms to sort case loads from data. Expectations of what machine learning technologies can do should be tempered by the caveat that research on the psycho-sociology of violent political extremism is itself work-in-progress (Smith, 2018), and the best methods for harnessing machine learning techniques to track complex human behaviors are in the early stages of development.

Recently, transfer learning has been gained a significant advancement in image classification. `Keras` [1], a Python machine learning library provides many pre-trained deep learning models that are made available alongside pre-trained weights for image classification (*Keras Applications*, n.d.). Furthermore, the transfer learning capabilities were already applied for text classification. BERT (Bidirectional Encoder Representations from Transformers) (Devlin, Chang, Lee, & Toutanova, 2019) is one of the best NLP models available with significant enhancements in the NLP domain especially having smaller and incomplete training datasets. BERT is designed to pre-train deep bidirectional representations from the unlabeled text by jointly conditioning the left and right context in all layers. BERT advances state of the art for eleven NLP tasks. BERT is a generic NLP library that allows customizing the last layers of the neural network to achieve various NLP tasks. Fundamentally, BERT excels at handling what might be described as 'context heavy' language problems. We used BERT for multi-label text classification for radicalization information extraction with the WJDB dataset and obtained significant improvement compared to other NLP approaches.

[1] https://keras.io/

2.3 Investigative graph search

Our work is on inexact graph pattern matching in graph databases while focusing on investigative applications involving social elements. Social network analysis has received significant attention and made important advances in recent years, largely due to the success of online social networking and media-sharing sites, and the consequent availability of diverse complex and heterogeneous social network data (Bonchi, Castillo, Gionis, & Jaimes, 2011). In many problem domains related to social networking data, exact match solutions do not provide insightful answers, e.g., assessing behavioral patterns for investigations, evaluating purchase patterns, studying disease patterns, and providing recommendations. Therefore, inexact matching techniques are beneficial, and such approaches allow finding subgraphs that nearly satisfy a given query graph.

The exact subgraph matching problem is a relatively easy task as we look for a given pattern. Ullmann's Algorithm (Ullmann, 1976) is known to be the initial approach for exploring isomorphic patterns based on query graphs in larger networks (Asiler & Yazıcı, 2017), and the structural pattern of the query graph. It iterates through all possible mapping nodes by performing depth first search algorithm while engaging various pruning techniques. Matching order strategies and effective pruning rules are further enhanced in VF2 (Cordella, Foggia, Sansone, & Vento, 2004), QuickSI (Shang, Zhang, Lin, & Yu, 2008), and GADDI (S. Zhang, Li, & Yang, 2009). Tree search-based algorithms are relatively effective for exact matches in big-data contexts. VF3 algorithm (Carletti, Foggia, Saggese, & Vento, 2017) introduces a novel subgraph isomorphism approach by ensuring the efficient performance on large and dense data graphs, and consists of depth-first search and backtracking including efficient heuristic rules to reduce the search space. *BB-Graph* (Asiler & Yazıcı, 2017) presents an exact subgraph matching approach with a branch-and-bound technique using graph databases. They present a novel algorithm using a Neo4j graph database and utilize database features to improve the search.

Inexact pattern matching techniques have also evolved with social network analysis, where vast networks of heterogeneous and labeled graph data are available. An incremental graph pattern matching algorithm is proposed in (Kanezashi, Suzumura, Garcia-Gasulla, Oh, & Matsuoka,

2019) to deal with time-evolving graph data with an adaptive optimization system based on reinforcement learning. A best-effort pattern matching technique for labeled graphs is proposed in (Tong, Faloutsos, Gallagher, & Eliassi-Rad, 2007), which aims to maintain the shape of the query. An inexact matching technique presented in (Hlaoui & Wang, 2002) based upon dissimilarities between the query graph and the data graph and then seeks to optimize a cost function by selecting matching nodes. The approach is computationally expensive for dynamic networks. The graph-querying framework NeMa proposed in (Khan, Wu, Aggarwal, & Yan, 2013) allows for ambiguity in both the structure and vertex labels. Instead of checking for graph isomorphism, NeMa identifies the top-k optimal matches by minimizing vertex labels differences and vertex pair distances. It is designed to return possible results that deviate from a given query pattern and is applicable when the query graph is not well defined. With two ranking schemas for matches, i.e., relevance and distance, another top-k pattern matching method is introduced (Fan, Wang, & Wu, 2013) that presents a generalized top-k matching function that couples both generalized relevance and distance functions. A top-k user-defined vertex scoring query method for edge-labeled graph databases is proposed with two scoring techniques (Parisi, Park, Pugliese, & Subrahmanian, 2018). An inexact matching technique was proposed in (Olmos, Gonzalez, & Osorio, 2006), which was uniquely defined as finding graphs with identical topology while allowing differences in vertices and edge labels. In contrast, we consider topology, node and edge labels, as well as node properties that need to be matched. The inexactness involves identifying a minimal number of matching nodes and edges. *Dual simulation* (Ma, Cao, Fan, Huai, & Wo, 2014) extends Ullmann's algorithm that searches for binary match relations between query and data graphs. It preserves both parent-child and child-parent relationships in the match and thus produces more meaningful matches.

Investigative simulation (B. W. Hung & Jayasumana, 2016) extends dual simulation to obtain inexact matching in isomorphic patterns. It produces matches of potential subjects that may require further investigation. This work proposes to detect the radicalization of homegrown violent extremists based on online and offline behavior, and produce a categorical node labeling mechanism by giving weight to each node based on its activity. An investigative framework for detecting

radicalization trajectories in large heterogeneous graphs (B. W. Hung et al., 2018) demonstrates the scalability of the approach in a large dataset. We build on (B. W. Hung & Jayasumana, 2016; B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019) and provide a more scalable investigative graph search via graph databases. In (B. W. Hung & Jayasumana, 2016; B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019), the data is stored in files which must be brought into the memory for processing. However, this is inefficient for dynamic data. We focus on maintaining a graph data store and leveraging its features to make processing of queries in real-time more efficient.

2.4 Complex object generation

Advanced data collection techniques and online social media platforms produce UDS in an extraordinary scale, and thus social network analysis can now be used to inform solutions to many societal issues (Bonchi et al., 2011). However, data integrity is a major concern in social networks as many fake and misleading data are not uncommon(Muramudalige et al., 2019). In many disciplines, such as economics, biological, and social sciences, removal of non-verifiable entries is crucial for maintaining the required data integrity, which in turn leads to UDS. Also, in many other applications of UDS, data values are unavailable because they were not measured, not known, or do not exist which is inherent in many social and behavioral domains (Klausen, Libretti, et al., 2018). Examples include behavioral patterns of specific individuals and groups in homeland security (Campedelli, Cruickshank, & Carley, 2019; Campedelli, Bartulovic, & Carley, 2019; B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019), suspicious network activities in cybersecurity (Peng, Xu, Xu, & Hu, 2017), recommendation systems in consumer analytics (Vassøy, Ruocco, de Souza da Silva, & Aune, 2019). Among applications related to healthcare and well-being that face the challenges we address are the behavioral patterns of patients to determine illnesses (Islam, Shelton, Casse, & Wetzel, 2017; Mancini & Paganoni, 2019), especially mental and suicidal prevention applications. Various techniques are introduced to handle missing data in different contexts (Folch-Fortuny, Villaverde, Ferrer, & Banga, 2015; MacNeil

Vroomen et al., 2016). However, there is no such technique to model and synthesize UDS data to the best of our knowledge. While the proposed method is applicable to a broad set of UDS, we present results for three specific applications interest to us are applications related to social networks, homeland security, and medical records.

With the proposed feature mapping technique, we use copula as a parametric modeling approach to synthesize UDS. Copula based likelihood approach enables modeling dependence structures for the distributions of dependent random variables (Ly, Pho, Ly, & Wong, 2019) and widely applied in econometrics (Cherubini, Gobbi, & Mulinacci, 2016), finance (Genest, Gendron, & Bourdeau-Brien, 2009), and risk management (Jammazi & Reboredo, 2016), especially in modeling financial risks (X. Zhang & Jiang, 2019), and diverse ranges of forecasting applications (Z. Wang, Wang, Liu, Wang, & Hou, 2017; Zhao, Wang, & Zhang, 2019; Panamtash, Zhou, Hong, Qu, & Davis, 2020). In many engineering applications, it is common to assume that the features are mutually independent or coupled by a Gaussian or elliptical dependence structure (Torre, Marelli, Embrechts, & Sudret, 2019). However, most of the cases we have to deal with complex distributions and vine copulas are capable of overcoming such limitations. Vine copulas are models of multivariate dependence built from simpler pair-copulas and the vine representation is sufficiently adaptable to capture complex dependencies (Torre et al., 2019). Therefore, we use both Gaussian and vine copulas in the likelihood analysis to determine the efficient modeling of unconventional data.

Generative adversarial networks (GANs) (Goodfellow et al., 2014) have become an alternative for data generation without extensive problem-specific theoretical foundation or empirical verification (Yan, 2019). The initial GAN architecture (Goodfellow et al., 2014) is capable of capturing the exact distribution of continuous and complete data but cannot be used for learning the distribution of discrete variables (E. Choi et al., 2017). Therefore, many derivatives of GAN architecture were proposed to deal with discrete distributions. Adversarial Autoencoder (AAE) (Makhzani, Shlens, Jaitly, Goodfellow, & Frey, 2015) is one of the deep-generative network, which is a probabilistic autoencoder that uses the GAN framework as a variational inference algorithm for both discrete

and continuous latent variables. AAE has proven its capability by applying in diverse ranges of disciplines: biology (Kadurin et al., 2017) to computer networks (Shirazi, Muramudalige, Ray, & Jayasumana, 2020), to obtain exceptional results. Therefore, we select the AAE as the deep-generative approach to compare the generated data with likelihood approaches.

2.4.1 Synthetic profile generation

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) use for data generation without requiring extensive problem-specific theoretical foundation or empirical verification (Yan, 2019). The original GAN architecture (Goodfellow et al., 2014) is capable of capturing the exact distribution of continuous and complete data but cannot be used for learning the distribution of discrete variables (E. Choi et al., 2017). The critical need to capture data distribution with discrete features in diverse application domains such as phishing, medical, crime data, etc. was fulfilled and proposed the Adversarial Autoencoder (AAE) (Makhzani et al., 2015), which is a probabilistic autoencoder that uses the GAN framework as a variational inference algorithm for both discrete and continuous latent variables. With a combination of an autoencoder and the adversarial framework, medGAN (E. Choi et al., 2017) is able to capture the distribution of discrete features, such as diagnosis or medication codes.

Other domains also need a good volume of high quality data. Scenarios involving social analytics (B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019; Muramudalige et al., 2019), privacy (Beaulieu-Jones et al., 2018), and health informatics are some areas that face the issue of limited data availability and data incompleteness. Data collection and maintenance are challenging because of data privacy and confidentiality issues. Behavioral and social network data are inherently sparse and incomplete because sometimes the behavioral indicators are not shown or recorded (Klausen, Libretti, et al., 2018). In this research, we proposed an adversarial data generation technique using sparse, incomplete, and small training samples. The method was validated via a domestic radicalization dataset which was a small and incomplete dataset.

2.4.2 Phishing websites detection

Machine learning algorithms are well suited for phishing detection as they can assimilate common attack patterns such as hidden fields, keywords, and page layouts across multiple phishing data instances and create learning models that can detect whether a given website is genuine or phishing. In the prior machine learning approaches (Niakanlahiji, Chu, & Al-Shaer, 2018; Sahingoz, Buber, Demir, & Diri, 2019; Mao et al., 2019; Jain & Gupta, 2018; Hong, Kim, Liu, Park, & Kim, 2020; Patil, Thakkar, Shah, Bhat, & Godse, 2018), researchers engineered novel sets of features from diverse perspectives using public datasets or their own generated datasets. The models were trained on phishing and legitimate datasets. These models were then used to predict whether unknown datasets are genuine or phishing.

PhishMon (Niakanlahiji et al., 2018) introduced a scalable feature-rich framework with a series of new and existing features derived from HTTP responses, SSL certificates, HTML documents, and JavaScript files. It does not rely on third-party services to extract features and is language agnostic and detects phishing instances in real-time. The authors reported accuracy of 95% on their datasets. 154 features were extracted based on the content of a webpage merging with four time-based, two search-based, and 11 heuristic features to create a labeled dataset (X. Zhou & Verma, 2020). Then, they created a balanced dataset with 8180 instances. Zhou *et al.*compared Random Tree as the best performing classifier among other classifiers and achieved precision of 99.4% and 0.1% false positive rate. Visual similarity of phishing and legitimate websites are studied by comparing Cascading Style Sheets (CSS) using an automated process (Mao et al., 2019). They proposed a learning-based aggregation analysis mechanism that can distinguish phishing websites from legitimate ones.

Detecting phishing instances by analyzing the URL of phishing websites have been widely studied in the literature. Sahinguz *et al.* (Sahingoz et al., 2019) proposed a set of natural language processing based features on URL of the websites and ran seven different classification algorithms to detect phishing websites. This study is language independent and can detect phishing websites in

real-time without needing third-party services. They achieve a 97.98% accuracy rate for detecting phishing URLs.

Shirazi *et al.*(Shirazi, Bezawada, & Ray, 2018) observed datasets used in the literature are inadvertently biased with respect to the features based on the website URL or content. Moreover, some of the features may become obsolete with time or as new attacks emerge. Sometimes the authors extracted features for the first page of legitimate websites, not the other pages. A machine learning algorithm will be useful if trained on enough data samples, but there is not a simple way to estimate the needed dataset size. The right size is related to the complexity of the problem and the complexity of the learning algorithm. This could be seen as a type of sample size determination (SSD) that evaluates the needed sample size in a specific problem.

For example, Figueroa *et al.* described a sample size prediction algorithm that conducted weighted fitting of learning curves in an active learning algorithm (Figueroa, Zeng-Treitler, Kandula, & Ngo, 2012). Active learning systems attempt to minimize the number of required labeled data and maximize the performance of the model by asking queries in the form of unlabeled instances to be marked by another agent such as the domain expert (Settles, 2009).

2.4.3 Video trace generation and classification

A plethora of works has been done on encrypted video traffic classification using ML approaches to address two main aspects: security & privacy of the users and network resource optimization. In terms of privacy, Li *et al.* (Y. Li et al., 2018) fingerprints closed set of YT videos, using the data collected from network and Media Access Command (MAC) layers. They leverage 3 types of neural network models—Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Multi Layer Perceptron (MLP) network—having more than 300 traces for each video. In (K. Choi et al., 2020), the authors extend the work in (Y. Li et al., 2018) to classify both content provider and video content based on a hierarchical model consists with XGBoost classifiers, which takes derived features from flow data. In order to to provide insights for effective resource allocation and traffic handling, (Dimopoulos, Leontiadis, Barlet-Ros, & Papagiannaki,

2016; Schmitt et al., 2019) analyze user QoE (Quality of Experience) taking flow level and packet level information where feature engineered data is fed to traditional ML classifiers. The same goal is achieved by DNN model feeding raw packet data in (Gutterman et al., 2019; Jiang et al., 2020). Apart from above 2 aspects, Rezae *et al.* (Rezaei & Liu, 2019) present recent trends in application of DNN in network traffic domain highlighting that DNN is more generalizable than traditional ML methods as they do not require tedious feature engineering efforts.

Before deep-generative techniques are available, probabilistic approaches like copula was used, which is a likelihood approach that enables modeling dependence structures for the distributions of dependent random variables (Ly et al., 2019) and widely applied in the finance sector (Genest et al., 2009). Some copulas are flexible high-dimensional dependence models that can capture non-Gaussian complex structures (Sun, Cuesta-Infante, & Veeramachaneni, 2019). With the introduction of the Generative Adversarial Network (GAN) architecture (Goodfellow et al., 2014), the data generation has achieved a remarkable improvement while capturing the underlying complex distributions. The ordinary GANs are fluent in capturing complete, consistent data distributions such as images (Karras, Laine, & Aila, 2019), audios (Donahue, McAuley, & Puckette, 2018), and videos (Clark, Donahue, & Simonyan, 2019). Later, the generative adversarial approaches for discrete distributions were introduced (Makhzani et al., 2015; Arjovsky, Chintala, & Bottou, 2017). Recently, GANs are being modified to generate time-series data in a diverse range of domains (Fathi-Kazerooni & Rojas-Cessa, 2020; Lin, Jain, Wang, Fanti, & Sekar, 2019). Therefore, GANs are widely applicable in many disciplines while addressing different data-specific challenges.

Chapter 3

Problem Statement

Identifying latent and emergent behavioral patterns of individuals and groups is crucial in homeland security, counterterrorism, and other investigative domains. Law enforcement authorities and counterterrorism professionals are in dire need of efficient computing tools to detect violent extremists in advance. Even though the related data is available on large volumes with the latest data collection methods, incompleteness and sparseness of the data mitigate applying current detection models and techniques. Analyzing latent and emergent behavioral patterns is a delicate issue that needs to address scalability and dynamicity in large, heterogeneous social and knowledge networks. Behavioral patterns analysis is beneficial in a wider range of domains such as cybersecurity, consumer analytics, and other disciplines where behavioral patterns play a significant role. Therefore, this work focuses on automating investigative pattern detection that depends on multiple tasks and requires diverse techniques and tools integrated with machine learning and graph pattern matching techniques.

Advancements in data collection and text processing technologies now allow collecting and aggregating people's profile information and behaviors. Such information is represented in the form of social and knowledge networks consisting of people's connections and behaviors. Social and political scientists have intensely studied the behavioral indicators of radicalized people and identified template profiles that may be used for classification of radicalization stages (Klausen et al., 2016) (Klausen, Libretti, et al., 2018). Such templates of individuals guide the detection of high-risk profiles in knowledge networks using various graph search techniques. To address these challenges, we propose an investigative pattern detection framework for counterterrorism integrated with different components and thoroughly discussed in Chapter 4. First, we automate steps for identifying radicalization behavioral indicators in text extracted from disparate text sources such as court documents, analyst notes, verified social media accounts using a set of natural language processing tools. The data extraction simplifies the process of forming large informative knowl-

edge networks. A persistent and scalable data store also enables maintaining the dynamic network data for a long time. Based on the comprehensive studies of social scientists (e.g., WJDB (Klausen et al., 2020)), a model graph can be implemented to represent the generic behavior of a suspicious person. Such a model graph is considered a query graph for inexact graph pattern matching. Investigative graph searches are introduced to explore potential risk individuals and groups on large knowledge networks. Furthermore, we propose a novel synthetic data generation technique to mimic the extremist profiles based on their behavioral indicators that aid social scientists to expand their studies and implement more robust evaluation tools.

The primary goal of this research is to automate the investigative pattern detection process by comprehensively studying the behavioral indicators of the extremists and implementing an end-to-end investigative framework for social scientists and law enforcement authorities. The proposed investigative detection framework significantly narrows down potential individuals and groups that require further investigations by law enforcement authorities. The proposed framework improves the efficiency of document inspection for radicalization indicators by social scientists while contributing to the human-in-the-loop (HITL) mode of operation to verify and validate the pattern detection process. HITL helps to iteratively share the multi-disciplinary knowledge between social and computer scientists, eventually implementing robust and scalable investigative pattern detection tools and techniques. While enhancing the efficiency of investigations, it filters a limited number of potential individuals and groups from an extensive knowledge network that may include billions of individual data. Ultimately, identifying risk profiles and early intervention will prevent radicalization and fatal attacks. With the proposed data generation technique for behavioral patterns, which are implicitly sparse and incomplete, we contribute to many other research domains that suffer from a lack of data records for robust and accurate evaluations and classifications.

3.1 Objectives

This research develops tools and techniques integrated into an end-to-end framework to efficiently and scalably to detect latent and emergent patterns in large-scale heterogeneous knowledge networks. To achieve our goal, we classify our specific objectives into the following categories.

3.1.1 Data extraction and NLP techniques

- Develop tools to automate the information extraction from text sources. Usually, this process is done manually by social scientists and coders and is considered a tedious and inefficient task

- Accelerate the efficiency of identifying behavioral indicators in disparate text sources and provide more structured methods to extract information

We integrate several NLP techniques to extract and identify behavioral indicators, notable features like timestamps and relationships. To extract related text, trustworthy text sources are utilized such as court documents, verified social media accounts, and analyst notes of other potentially risky people. As the first step, we use NER (named entity recognition) to classify person names, organization names, date and time, etc. Coreference resolution is utilized to identify similar clusters of the subjects and objects of the text. Rule-based matching, which is an annotation method for finding specific patterns of tokens in the text, is utilized to detect linguistics patterns of behavioral indicators. It allows implementing a set of custom rules to identify relevant and unique text patterns. Furthermore, Multi-label text classification, a supervised machine learning model, is implemented to classify sentences based on radicalization behavioral indicators. Multi-label text classification outputs the probability values for each trained indicator (label), implying the sentence's relevance to the multiple labels. We further use a transfer learning technique, a pre-trained NLP model called BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), for multi-label text classification by fine-tuning the BERT model with our comparatively smaller labeled dataset. The further details are described in Chapter 5.

3.1.2 Investigative graph search

- Implement information and knowledge networks to capture the extracted information from behavioral indicators while having an appropriate data storage to store these types of heterogeneous networks

- Develop algorithms to detect latent and emergent patterns such as those of individuals and groups of interest in social and knowledge networks using efficient and scalable graph pattern matching techniques

Large knowledge networks constructed with behavioral patterns are arranged after extracting behavioral information in text materials using various natural language processing techniques. In the long run, maintaining behavioral data is challenging due to the dynamicity and velocity of the data. So, it is essential to have a convenient data store to overcome these obstacles and provide scalable and efficient analysis. We use graph databases to store knowledge networks because they facilitate keeping the data as complex heterogeneous graphs. Then, complex investigative graph searches are required to detect potential individuals and groups. The generic graph template (query graph) of a suspicious person can be evaluated based on social and political scientists' extensive studies of radicalization behavioral indicators. An investigative graph search is more complicated because there is no fixed way or a pattern to exhibit behavioral indicators in any domain. Therefore, we implement an inexact, similarity-measure-based graph search on top of graph databases. The implemented investigative graph search can find similar patterns based upon a query graph in a given database. The graph search also contains a novel mechanism to detect potential suspicious groups that collectively satisfy a given query graph. We also verify the scalability and efficiency of the proposed graph searches with different datasets. The more details are presented in Chapter 6.

3.1.3 Synthetic profile generation

- Implement data generation scheme to synthesize radicalization profiles to overcome the limitations due to small set of documented profiles so that allow machine learning and classification techniques may be developed.

- Extend and enhance the data generation technique as a formal method to use in different domains which suffer from lack of adequate data.

- Apply, demonstrate and evaluate the synthetic data generation method to other applications with small and sparse datasets so as to enhance the performance of machine learning techniques.

Only a limited number (hundreds) of actual biographies and trajectories of radicalized persons are available to study and analyze (Klausen et al., 2020). Having a smaller dataset restricts a comprehensive analysis and implement precise detection models. Further, synthetic data may help address confidential and privacy issues due to sensitive information. Moreover, a sufficiently large dataset will enable training neural networks to capture the features of radicalized or suspicious people's behavioral indicators and provide a platform to study and model extremists extensively. A novel approach using generative adversarial training is proposed to generate extremist profiles while proving the statistical similarity of the generated results. The general details of the proposed data generation technique and radicalized profile generation are presented in Chapter 7 and 8, respectively. The proposed profile generation technique is applicable for discrete and sparse datasets in any other discipline. Therefore, we contribute to the other research areas that lack data due to various data collection challenges. The details of the other data generation applications are discussed in Chapter 9 and 10.

Chapter 4

An Investigative Pattern Detection Framework for Counterterrorism

4.1 Introduction

Counterterrorism professionals and law enforcement authorities have been searching for proper and efficient computing tools to identify violent extremists. Due to their dynamic and complex behavioral patterns, analyzing and modeling these events have been extremely complicated. After the 9/11 attacks in 2001, and then after 2015, there was a significant enhancement of the violent extremist attacks worldwide, which led to an increase the terror in many countries. Tracking and surveillance the behavior of billions of people is unrealistic for investigative authorities with computing efficacy and scalability. The robust and accurate detection is essential to get rid of false accusations that will obliterate the trustworthiness of administrative bodies. The FBI too mentioned the information problem that they dealt with thousands of queries daily (R. E. Travers, National Counterterrorism Center, 2019). These massive volumes of data are infeasible to process without having effective and reliable tools. Therefore, there is a critical need for a set of computing tools to process such data in supporting analysts in these assessments and finally enabling law enforcement to prevent attacks in advance.

However, analysts face significant challenges with respect to both discovery and knowledge management (Bellutta et al., 2020). The first is simply scalability–how to produce salient risk assessments that are derived from the sheer volume of disparate types of data (Hoffman, Meese, & Roemer, 2015). The data available are often found in disconnected portals, consisting of uploaded text reports from the field and not structured such that they can be fused, normalized, and co-referenced to enable the analysis needed for these risk assessments. The other significant challenge is dynamics–the dynamical nature of the threat as well as individual behavioral indicators

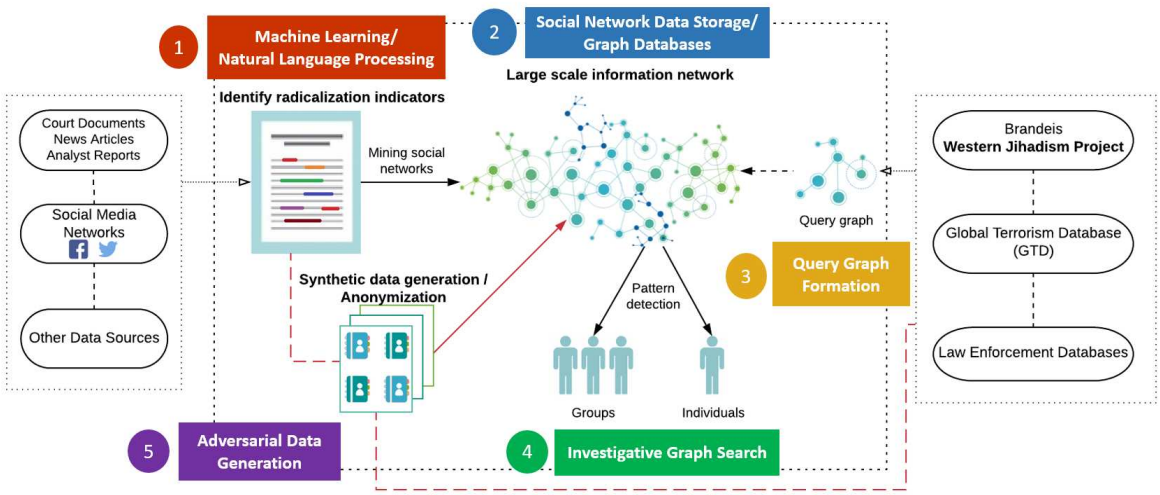


Figure 4.1: High level architecture INSPECT (Investigative Pattern Detection for Counterterrorism).

necessitate databases to be updated very frequently (Capellan & Lewandowski, 2018), which in turn requires more automated analysis capabilities, to include scaling and speeding up structured information extraction related to radicalization indicators, as well as fusing knowledge such that it is relatable and contextualized.

Machine learning techniques support the extraction of knowledge and predictions from large diverse digital data and are routinely used for predictive purposes. Such methodologies have proved useful for the extraction of knowledge from diverse digital data and predictive behavioral and social risk modeling in areas ranging from child abuse to extremist Twitter-users' adaption to having their accounts shut down. In this work, we introduce novel investigative pattern-detection framework techniques that rely on behavioral pattern detection of extremists from text and relational data to flag cases and aid a human analyst to sort signal from noise. To be clear, we are not dealing with metadata used for the purpose of mass surveillance but an effort to use machine learning to extract risk profiles from an evidence-based sociological model of human behaviors known to be associated with escalating risk of violent action (Freeland et al., 2019).

The framework consists of several computing tools in the process of automating the investigative pattern detection through machine learning and graph pattern matching techniques. The value of the combination of a machine learning approach to text mining fused with a validated socio-

logical model of radicalization with network science is the ability of computational methods to detect behaviorally meaningful signals from large amounts of data at great speed. The sociological data used to profile extremist behaviors used here derives from multi-year study of domestic jihadists by one of the authors. The proposed techniques in the framework may be adapted to other investigative detection contexts with appropriate datasets. Collaboration between social scientists and computer scientists is critical to the effort to build evidence-based machine learning models of complex human behavior. In such collaborative research, human-in-the-loop (HITL) plays a key role that allows domain expertise (social scientists) to validate the reliability and accuracy of the proposed investigative pattern detection framework by computer scientists. Further, HITL is an iterative process to improve the capabilities of the framework while sharing the knowledge and the experience between social and computational contexts.

The use of behavioral markers to profile extremist radicalization has been subject to controversy on privacy ground but is now widely accepted by social scientists and law enforcement (Lindekilde, O'Connor, & Schuurman, 2019). (Radicalization refers to the process by which people come to support violent extremism and join terrorist groups or commit a terrorist act.) Guidelines issued by the Office of the Director of National Intelligence with collaboration by the FBI, the US National Counterterrorism Center, and the US Department of Homeland Security, enumerates a list of observable indicators of potential violent extremists drawn from research. A study released by the FBI in 2019 of 52 ideologically-motivated lone offenders concluded that "they traveled down the same observable and discernable pathways to violence as other attackers" (Behavioral Threat Assessment Center of Federal Bureau of Investigation (FBI), 2021). The behavioral model of radicalization used here was developed in previous research by the authors (Klausen, Libretti, et al., 2018).

To address the critical needs of scanning and mining large volumes of data while adapting to scale and dynamicity and to support human-in-the-loop investigative searches we propose an end-to-end investigative pattern detection framework INSPECT (Investigative Pattern Detection Framework for Counterterrorism). Natural Language Processing (NLP) techniques are integrated

and adapted in INSPECT to extract radicalization behavioral indicators and features (such as timestamps, relationships) in different text sources. The extracted data are arranged to a knowledge network consisting of behavioral patterns of individuals of interest. Machine learning and graph pattern matching techniques are used for investigative graph search proposed to identify individuals and groups from the large information and knowledge networks. The solution is built on top of a graph database, with its convenient data storing mechanism, to achieve the efficacy and scalability for dynamically manipulating large, heterogeneous networks. Further, we introduce a novel synthetic profile generation technique for behavioral patterns to address two intrinsic challenges in the domain:

1. the lack of sufficient training data for both human coders (whose expertise pertain to identification and classification of indicators in documents) and machine learning models

2. The need for shareable anonymized data sets of different sizes and characteristics that do not violate various privacy regulations or constraints.

4.2 INSPECT architecture

The overall architecture and components of the INSPECT framework is illustrated in Figure 4.1. It consists of the following major functional components:

1. *NLP to identify radicalization indicators in text sources:* The details of extremists are mostly available as news articles, court documents, and reports, which are disparate text sources. The data consists of various behavioral indicators in different stages of radicalization. We apply several NLP techniques to extract the behavioral indicators and other information.

2. *Graph databases:* The extracted data from text sources are innately captured in the form of a social network that consist individuals together with their behavioral indicators, as well as links connecting individuals, organizations and behavioral indicators. This dataset is highly linked and storing and processing such data is a challenging task. Graph databases, designed

for pattern-based querying over huge volumes, contains many features for mining such networks. Therefore, we use a graph database over SQL and NoSQL databases where considers the relationships (links) between data as equally important as the data itself.

3. *Query graph formation:* With years of experience observing the extremists' behavior, social scientists have studied the diverse patterns of radicalization. We use their empirical knowledge to model query graphs representing the ordinary or specific behavior of an extremist. The example used here draws in the relational ontology developed by one of the authors assess contagion networks in a jihadist population.

4. *Investigative graph search:* We have developed and implemented a set of algorithms to explore potentially risky individuals and groups on knowledge networks (Muramudalige, Hung, Jayasumana, Ray, & Klausen, 2021). Graph searches are performed as custom queries to the graph databases, which enhances the efficiency and the scalability of data processing while utilizing the database features.

5. *Adversarial data generation:* We propose a novel synthetic data generation technique to mimic the behavior of extremists. This method is widely applicable in other domains as well where the available datasets are small, sparse, or insufficient.

4.2.1 Behavioral indicators extraction in text sources

Investigating the behaviors of already-identified violent extremists is a crucial instrument for recognizing emerging profiles based on similar behavioral patterns. Most such data is available in public domain text sources, which may be mined to study their behavioral nature without violating confidentiality restrictions. Social scientists rely on official press releases, court documents, trusted news sources and verified social media accounts of extremists. The common practice is for researchers to read and inspect related text documents and then manually label the behavioral indicators present (a process known as 'coding'). The training dataset and text cues used for the research and to train NLP algorithms was provided by the social scientists at the Western Jihadism

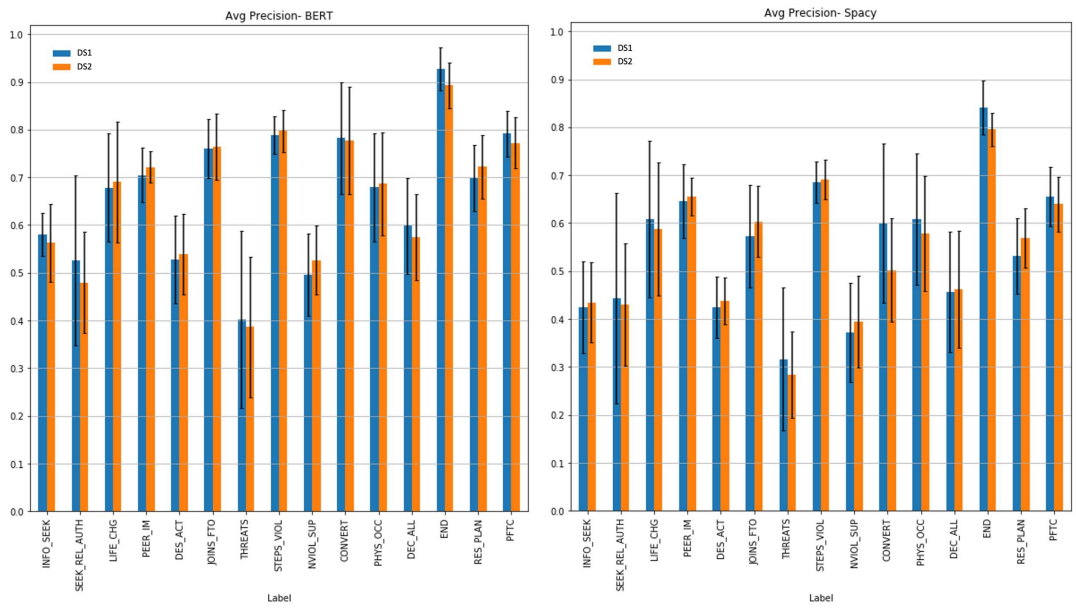


Figure 4.2: Each indicators' classification precision across 10-folds for both BERT and SpaCy. x-axis represents 15 different behavioral indicators (labels). Two training datasets are shown; DS1 (blue) and DS2 (orange).

Project (WJP), and contains phrases, sentences or short paragraphs which were manually labeled with up to 3 radicalization indicators.

Among different text classification techniques, we have made significant improvements using multi-label text classification (B. W. Hung, Muramudalige, et al., 2019), a supervised learning model where text documents are assigned one or more trained categories/labels. We focused initially on a sentence-level classification for simplicity and a finer analysis granularity for social scientists. In this work, 15 radicalization indicators are trained on a deep-classification model where returns probability scores for all labels (indicators) in each sentence in the document. We use SpaCy library (a convolutional neural network for NLP) and BERT (a pre-trained transfer learning model for different NLP tasks).

Due to the unbalanced nature of the training data (where some labels are more prevalent than others), a stratified train-test splitting was implemented to ensure that the resulting test set has a proportional set of labeled data akin to the training set as well as 10-fold cross validation.

Figure 4.2 depicts the precision scores with 10-folds for each of the labels for both the BERT and SpaCy models using two recent training data sets. Also displayed are the inter-label stan-

dard deviations (as error bars) and the intra-label standard deviations (shown below the plots). In both datasets, `BERT` outperforms `SpaCy` because of `BERT`'s ability to distinguish underlying data classes as well as a pre-trained model on a diverse range of generic corpora, even though our datasets are inherently unbalanced and small. Furthermore, the classification results were sent to social scientists for validation as a part of the HITL process. Based on the model results, social scientists fine-tuned the training set by understanding the issues in the dataset. Human validation is essential in the law enforcement domain to produce accurate results with sensitive behavioral data. Therefore, we have gone through multiple iterations to fine-tuned both the training dataset and the NLP models in the HITL process.

Our implementation also integrates the Named Entity Recognition (NER), an information extraction technique to detect pre-defined entities in text sources as person names, organizations, locations, time, etc. We use the SpaCy NER module[2] to detect date, time, person names, and organizations. In this case, NER enhances the capability to find relevant metadata of extremists in diverse text sources, which in turn are embedded while forming knowledge networks. Coreference resolution is another NLP task to find all expressions that relate to the same entity, which we use to extract the similar expressions related to an individual. However, our investigations revealed that the dynamicity and different lengths of the text sources led erroneous results. We were able to improve the results using different text preprocessing techniques such as lower casing, stop words removal, lemmatization. With these NLP tasks, we extracted the relevant information, including behavioral indicators and their relevant metadata, to produce structured knowledge networks. An important observation is the fact that in the HITL context, these NLP outcomes significantly help social scientists improve their manual coding process. These enhancements improve their efficiency allowing them to process considerably higher volumes of text sources while magnifying the knowledge of trajectories, leading to a comparatively larger dataset to train more robust NLP models.

[2]https://spacy.io/usage/linguistic-features#named-entities

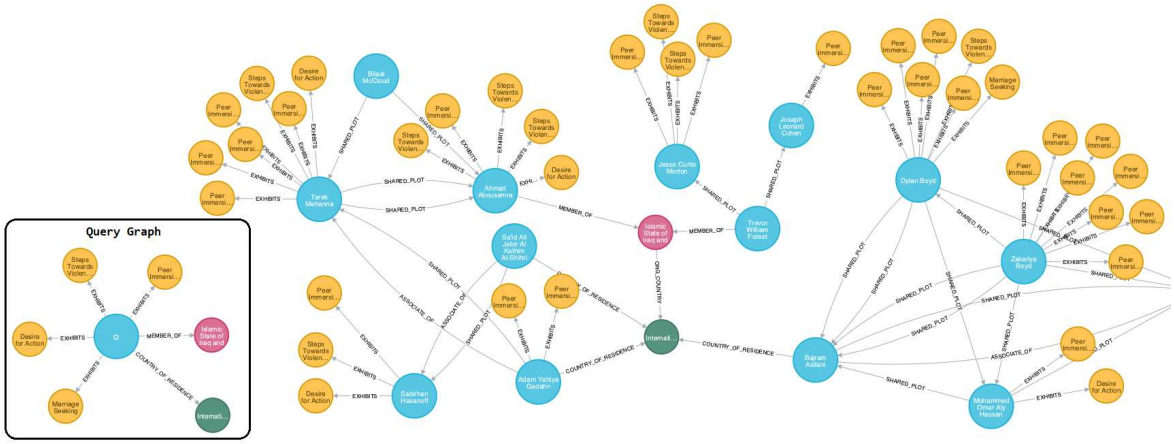


Figure 4.3: A subset of results for an inexact match *(similarity threshold = 0.7)* neighborhood measure in the WJP graph database based on a given query graph. The highlighted box represents an example query graph. The blue node represents an individual that presents in the network. Yellow nodes represent the behavioral indicators with their types. Green and red nodes depict a country and a terrorist organization, respectively.

4.2.2 Investigative graph search of knowledge networks

Identification of structured information of behavioral indicators via NLP tools and techniques allows for the formation of an enriched knowledge networks of individuals while dealing with the dynamicity and scalability of behavioral patterns. Moreover, persistent storage necessitates storing such time-critical networks for a long term. We use a `Neo4j` graph database to store knowledge graphs while adapting to the dynamicity and scalability of the behavioral data. Neo4j is currently the foremost graph database platform and contains various useful features to implement custom graph searches in a specific domain while enabling the consumption of basic graph algorithms (Muramudalige et al., 2021).

An investigative search to detect potential extremists or groups in knowledge networks in the `Neo4j` graph database complements the pattern detection and analysis. INSPECT relies on a similarity measure based inexact graph pattern matching technique that we proposed in (Muramudalige et al., 2021) to investigative individuals and groups. The method addresses the inexact sub-graph isomorphism problem because the similarity measure executes w.r.t a given query graph. The query graphs used in these experiments are based on the relational ontology developed by one of the authors assess contagion networks in jihadist populations.

An investigative search retrieves a set of suspicious individuals or groups. To this end, our system integrates a vectorized, similarity-based solution approach called INSiGHT (B. W. Hung et al., 2018) for investigative graph search. With the understanding of the reliability and scalability, we then developed more advanced investigative graph search library that runs on top of the `Neo4j` database with custom procedures was developed: *similarity measure* (for individuals) and *neighborhood measure* (for groups) (Muramudalige et al., 2021). We further discuss the extensive details of the investigative graph search, including the scalability of this approach with database features.

The identification of suspicious groups is crucial in law enforcement domains. Radicalized extremists are known to conspire with individuals or groups with similar interests for specific objectives. Searching for single-person subgraphs is therefore not sufficient to detect a more complete risk assessment that includes collectively suspicious behaviors. Investigators believe, e.g., that the plot in San Bernardino Terrorist Attack in 2015 was not detected before the attack because the perpetrators were radicalized separately before meeting each other. When detecting multiple individuals as a group, their collective behavior could have a magnifying effect in sudden extremist development or a threat of an attack. To identify such plots, it is vital to detect associations and collective behaviors via the proposed neighborhood measure. Figure 4.3 depicts a subset of the inexact neighborhood measure results in the WJP graph database based on the query graph shown in the highlighted box that represents an example query graph with four different indicators (orange) and two filtering options by the country (green) and the organization (red). These graph search techniques allow detectives and relevant authorities to focus on a smaller subset of individuals who have a higher likelihood of being extremists, rather than surveilling an enormous number of people, allow them to take necessary actions efficiently within a smaller time frame to avoid such fatal attacks.

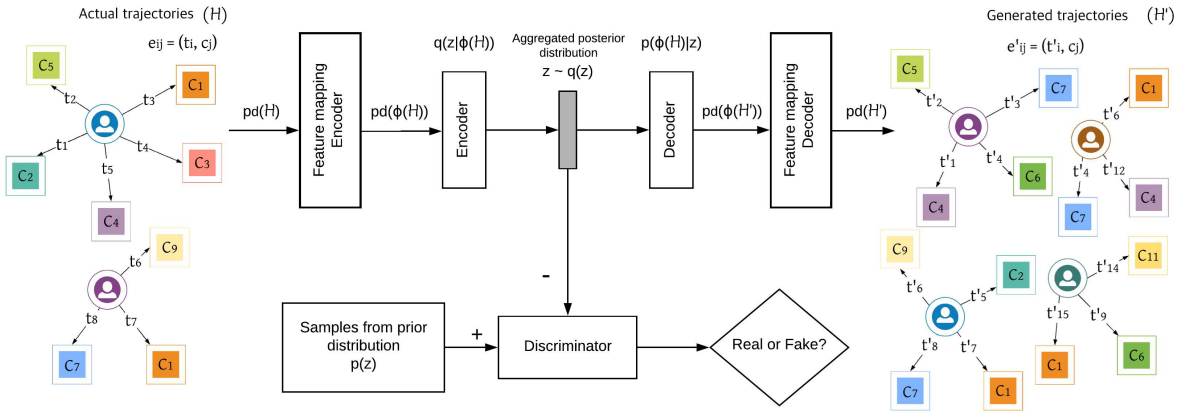


Figure 4.4: Synthetic data generator architecture for INSPECT consisting of an ordinary adversarial autoencoder couples with the proposed feature mapping encoder and decoder. The top row depicts the autoencoder that reconstructs the feature-mapped data from the latent code z. The second row shows the discriminative network that predicts whether the samples emerge from the hidden code of the autoencoder $q(z)$ or the user-defined prior distribution $p(z)$. $p_d(\mathcal{H})$ and $p_d(\phi(\mathcal{H}))$ denotes the actual and the feature-mapped data distributions, respectively. $p_d(\phi(\mathcal{H}'))$ denotes the generated feature-mapped data distribution. $p_d(\mathcal{H}')$ denotes the generated data distribution after send through the feature-mapped decoder. $q(z|\phi(\mathcal{H}))$ and $p(\phi(\mathcal{H})|z)$ denote the encoding and decoding distributions of the autoencoder respectively.

4.2.3 Synthetic profile generation

Having only a very limited number of historical profiles of radicalized individuals, and thus a limited number of trajectories available are major hindrances in the evaluations and studies related to radicalization detection. Even though many deep-learning techniques may be applicable to help identify latent trends in behavioral patterns, such machine learning models still require large datasets for robust and realizable outcomes. Having smaller number of data limits the knowledge of human coders who manually annotate the text documents. A critically important requirement therefore is a data generation technique that generates realistic datasets starting with small and discrete datasets, which is beneficial in various domains. Further, synthetically generated trajectories provide a significant degree of anonymization with respect to original data, and therefore provides an important source of shareable and publishable behavioral data without violating any privacy regulations. Thus synthetic data generation described below is an integrated part of INSPECT.

We propose an adversarial approach that augments adversarial autoencoder (AAE) (Makhzani et al., 2015) with novel feature mapping techniques. The architecture of our proposed technique is depicted in Figure 4.4. We consider a behavioral indicator (e_{ij}) is consist of a timestamp (t_i) and

the category (c_j), which is the common in radicalization data. Irrespective of the data is inherently small and spare, the proposed feature mapping techniques allows to mitigate such challenges fluently. Feature mapping techniques utilize a cumulative distribution of each category to map the data to a consistent data space.

We have already demonstrated the effectiveness of the data generation technique for enhancing machine learning tools for other domains characterized by very limited data availability and/or inconsistencies in data. Synthetic data so generated have led to significant improvements in phishing websites detection, and video traffic classification. Therefore, the proposed approach is widely applicable. In radicalization detection, synthetic data generation is extremely useful to improve the algorithms, and also to train computer and social scientists on different aspects of radicalization.

With all these components in the proposed investigative pattern detection framework, we have also verified the framework's robustness and efficiency over diverse datasets. Some of the library routines are publicly available in `https://github.com/cnrl-csu`. We have demonstrated that the NLP techniques, the storage selection, and investigative graph search were implemented with the consideration of reliability and scalability. Further, the proposed modularized architecture of INSPECT (Fig. 4.1) maintains the generalizability of the detection process that allows us to add/update/disable different computing components based on the data and the requirements. Therefore, the proposed framework is capable of grasping diverse datasets and adapting any new or future technologies.

Chapter 5

Behavioural Indicator Extraction using NLP techniques

5.1 Introduction

Modeling and predicting extremist behavior is an overwhelming task for administrative bodies and counterterrorism officials due to dynamicity and the lack of consistent data. Among the operational shortfalls that prevent these administrators from achieving required success is the difficulty in dynamically assessing individualized violent extremism risk at scale given the enormous amount of primarily text-based records in disparate databases/data sources (MITRE, 2021). Analyzing the information of existing violent extremists in text documents (e.g., news articles, analysis reports, court documents, etc.) using well-defined behavioral indicators allows investigating behavioral models extensively. A booklet issued by the Office of the Director of National Intelligence (The office of director of national intelligence, 2019) with collaboration by the FBI, the US National Counterterrorism Center, and the US Department of Homeland Security, describes the lists of observable indicators of potential violent extremists. According to the definition of a homegrown violent extremist (HVE) in the booklet, is a person who advocates, engages in, or is preparing to engage in or support terrorist activities in furtherance of a foreign terrorist organization's objectives, but who is acting independently of foreign terrorist direction (The office of director of national intelligence, 2019). Furthermore, the complex behavioral patterns of homegrown extremists make the investigation more challenging. The current behavioral indicators are sometimes individually innocuous since it requires further investigation and surveillance to detect links between other persons to ensure the violent extremist activities and networks.

The behavioral information is available in erratic data storage, and researchers have to deal with text irregularities. Therefore, data extraction is a tedious task that leads to one of the key

challenges in extremist behavioral studies. Thus, human-in-the-loop (HITL) plays a significant role in collecting data in a well-defined format. The common practice is for social scientists to read and investigate related text documents and then manually label the behavioral indicators present (a process known as 'coding'). The coders manually extracted a core set of sentences and sentence fragments used to create a labeled dataset to train machine learning models. Coders were trained to read various publicly accessible documentation for evidence of the 24 distinct behavioral indicators theorized to be associated with radicalization and instructed to record the dates at which such behaviors were publicly observed. This analysis enabled the retrospective estimation of timelines for the radicalization trajectories. 24 behavioral indicators are depicted in Table 5.1.

ID	Name	ID	Name
1	Convert Date	13	Dawa-Real Life
2	Disillusionment	14	Epiphany
3	Trauma	15	Peer-Immersion
4	Personal Crisis	16	Phy./Dom. Training
5	Seeking Information	17	Marriage Seeking
6	New Authority Figures	18	Societal Disengagement
7	Rebellion	19	Desire for Action
8	Lifestyle Changes	20	Passive Support
9	Educational/Occupational Disengagement	21	Joins Foreign Organization
10	Drop-Out Date	22	Issues Threats
11	Underemployment	23	Steps towards Violence
12	Dawa-Virtual	24	Date of Criminal Action

Table 5.1: Behavioral indicator names collected by WJDB

To date, the coders have annotated over 3000 sentences or paragraph samples extracted from over hundreds of different primary sources and secondary sources such as the Western Jihadism Project codebook and the terrorist profiles summarized in [24]. Because a significant number of sentences or paragraphs refer to two or more indicators, the coders constructed a training dataset of over 3000 labeled sentences or paragraphs. The results of the implemented models are further validated by social scientists. Complex behaviors, linguistic winks, and slang present significant challenges to detection algorithms, including machine learning modeling. In this work, we have

applied various NLP algorithms for data extraction and iteratively validated by social scientists to enhance the robustness and accuracy of the detection algorithms as the part of HITL process.

5.1.1 WJDB dataset

The dataset was extracted from Klausen's Western Jihadism Database (WJDB) (Klausen et al., 2020), a collection that includes information on approximately 6,600 individual jihadists of Western origin or residence who have engaged in criminal terrorist action. All the data derives from public sources ranging from court records, government press releases, and autobiographical statements made by the terrorism offenders themselves on social media or in jihadist forums. The size of the dataset is still not sufficient to train any traditional machine learning model because the training dataset is comparatively small and diverse. In the current project, detailed forensic biographies were developed for 122 homegrown terrorism offenders who radicalized between 2001 and 2018 and committed terrorism-related crimes in this period. We discuss each NLP approach we applied to detect and extract relevant information of extremists in the following.

5.2 Named entity recognition (NER)

Named entity recognition (NER) is the task of identifying text spans that mention named entities and classifying them into predefined categories such as person, location, organization, etc., based on pre-trained NER models (J. Li, Sun, Han, & Li, 2020). NER follows the fundamentals for various natural language applications such as question answering, text summarization, and machine translation. Although early NER systems successfully produced decent recognition accuracy, they often require much manual effort in carefully designing rules or features (J. Li et al., 2020). In recent years, deep-learning-based NER pre-trained models have shown significant advancement in the NLP domain. Therefore, we apply the latest pre-trained NER model[3] to investigate the detection of different entities related to extremist data extraction. We use different text documents

[3]https://spacy.io/api/entityrecognizer

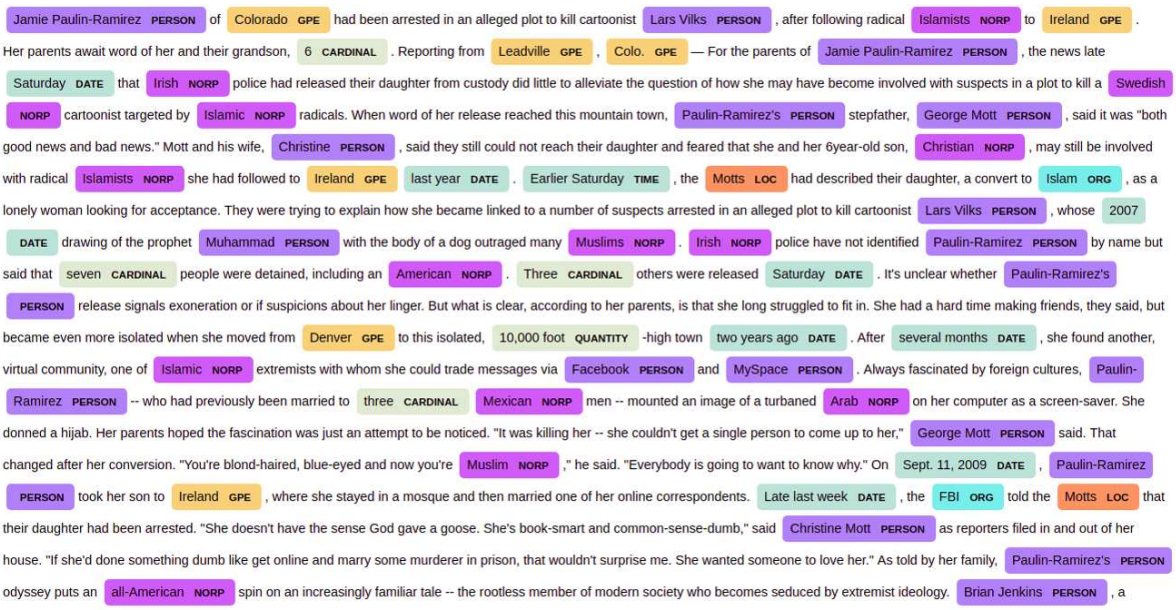


Figure 5.1: NER results for a news article

to extract person names, locations, and organizations and one of the output files related to a news article (The Associated Press, 2021) shown in Figure 5.1.

As part of this work, we train custom NER classifiers (e.g., introducing a *Radicalization* entity) to highlight words/phrases related to radicalization. However, with the currently limited training dataset, we cannot make sufficient progress to detect our custom entities. We also figured out many other limitations; the inability to allow for a specific keyword or keyword phrases associated with more than one indicator. Further, NER classifiers cannot maintain the contextual information of the text as NER detects entities independently by only examining a single word or phrase. Such shortcomings led to critical errors in the entity detection and limited NER classifiers to the document coders.

5.3 Coreference resolution

Coreference resolution is the process of grouping mentions/expressions into entities. A key challenge in this NLP task is that information about an entity is spread across multiple mentions. Therefore, deciding whether assigning a given mention to a candidate entity could require entity-level information that should be aggregated from all mentions. Coreference resolution is

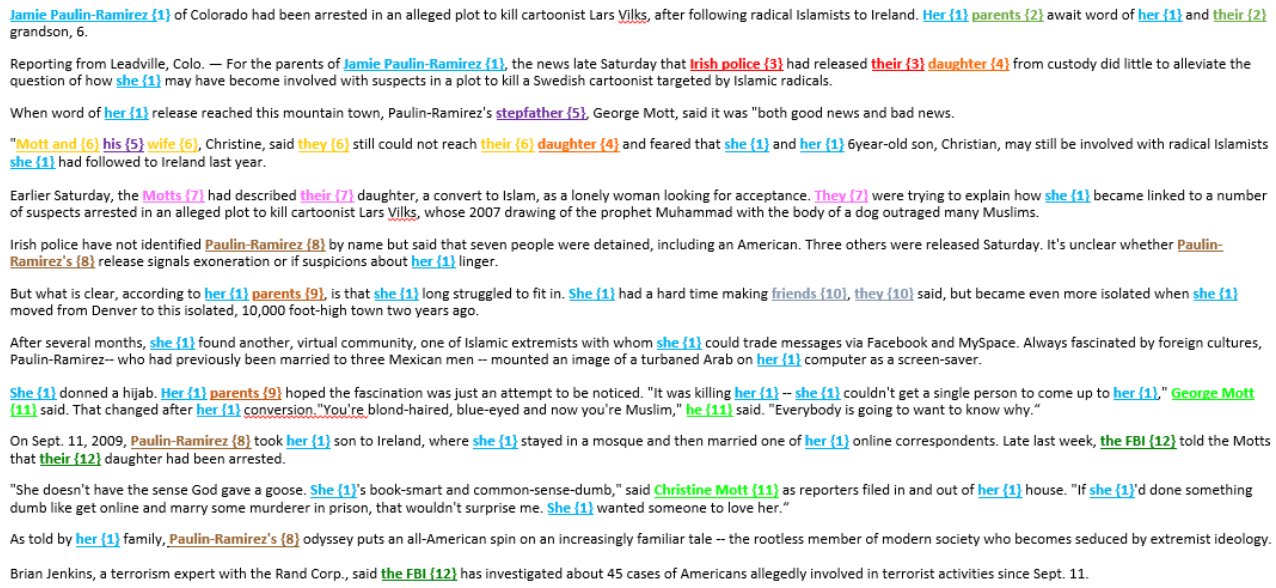


Figure 5.2: Coreference resolution results for a news article

an advanced approach to finding relevant entities compared to Named Entity Recognition (NER) because it evaluates the contextual information that spreads across the given text rather than assessing them independently. However, most coreference resolution schemes rely on entity mentions pairwise scoring, which is likely to miss the global information. In this work, we utilize the NeuralCoref[4] library built on top of the Spacy NLP pipeline. Figure 5.2 depicts the results for the same text applied in Section 5.2. The same color interprets the same mentions in the text.

We recognize its capability of identifying the same mentions to a certain extend. However, there are still some limitations. When there are multiple different mentions within fewer text ranges, the accuracy of the mentions significantly goes down. Also, when similar mentions are in a wide range of the text, there is a higher chance of missing some of the entities related to its incapability to capture global contextual information. However, recently introduced state-of-art NLP models like BERT produce significant improvements towards coreference resolution (Joshi, Levy, Weld, & Zettlemoyer, 2019). In the Section 5.5, we discuss applying the BERT model for multi-label text classification even with an imbalanced label training dataset.

[4]https://github.com/huggingface/neuralcoref

5.4 Rule-based matching

Rule-based matching is an annotation method for finding specific patterns of tokens in text. Rule-based matching is a next-level of keyword matching and still requires the manual production of an extensive set of rules based upon key word phrases from the training dataset. `SpaCy` NLP library provides a robust rule matching engine[5] while allowing to implement any number of rules. The following rule example consists of 3 tokens.

['LEMMA': 'watch', 'POS': 'ADJ', 'OP': '', 'LEMMA': 'video']*

The 'LEMMA' keyword provides the lemmatization and returns the base or dictionary form of a word. Rule-based matching enables the standardization of different tenses of verbs and plural words. The middle token expects an adjective, and 'OP' key defines it optional and allows an adjective zero or more times. Following are some of the word combinations that can capture from this rule.

- *watching jihadist videos*
- *watched a video*
- *watches terrorism related videos*

Therefore, The rule-based approach addresses the linguistic variation to a certain extent. However, the rule-based methods were not viable for identifying numerous and complex linguistic markers because of the difficulty for the researchers to scale the creation of a correspondingly robust ruleset.

5.5 Multi label text classification

The multi-label text classification is a promising NLP approach in our particular information extraction problem from extremist-related text documents because these texts are potentially assigned one or more categories or labels (Hung et al., 2019). As in the HITL process, the training corpus is obtained from the social scientists from WJP, contains phrases, sentences, or short

[5]https://spacy.io/usage/rule-based-matching

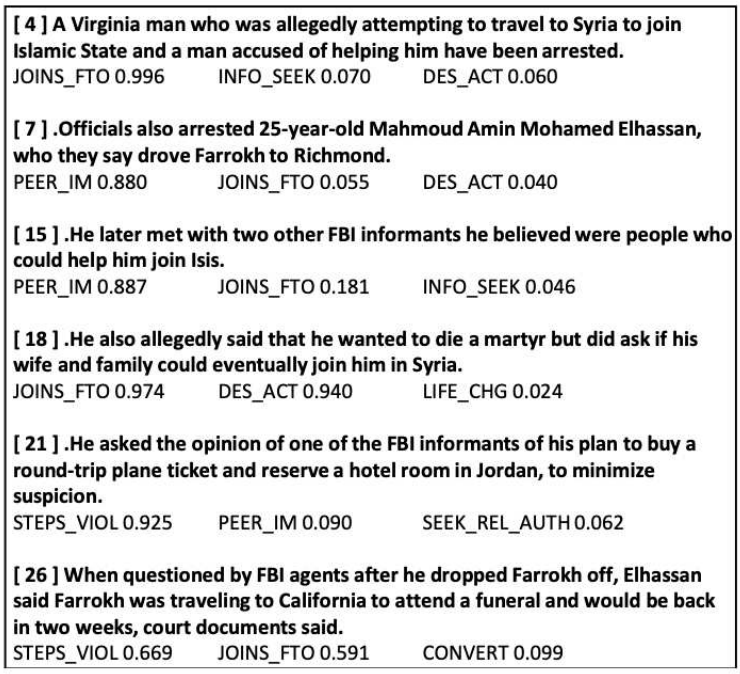


Figure 5.3: Sample output of multi-label classification model

paragraphs, which are manually labeled with up to 3 radicalization indicators. The corpus then undergoes standard pre-processing, which includes lemmatization and the removal of unique characters and stopwords. Initially, we implemented a workflow that used Prodigy[6], an text annotation tool that supports annotating text as multi-labels to generate training data for spaCy convolutional neural network (CNN) models for text classification. We focus on performing sentence-level classification to best aid law enforcement and intelligence analysts with the appropriate detail and classification granularity for specific behaviors. Initially, we train models with 10 behavioral indicators/labels where the multi-label classification model returns probability scores for each of the 10 indicator classes. Then, we post-process the results to obtain only 3 labels with the highest probabilities among 10 labels beneficial for further model accuracy and robustness evaluations. The sample output of the model is depicted in Figure 5.3.

The results files are sent to social scientists for their analysis. They thoroughly study the results and updated them about issues in the trained model. At the same time, they identify the

[6]https://prodi.gy

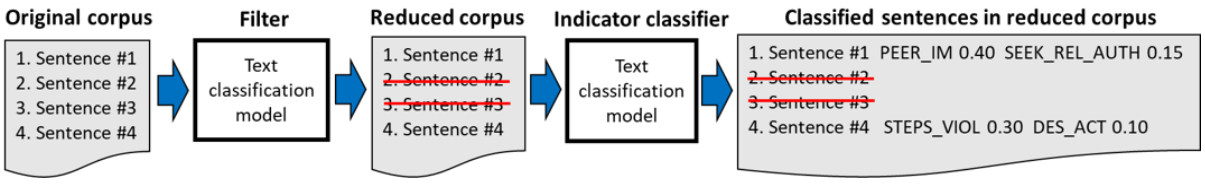
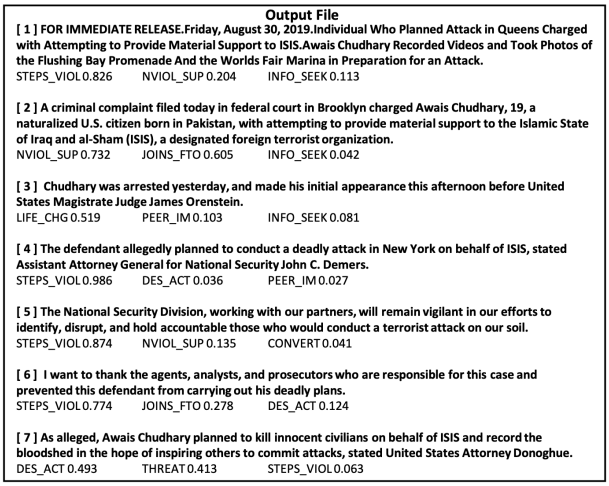


Figure 5.4: Multi-label text classification pipeline

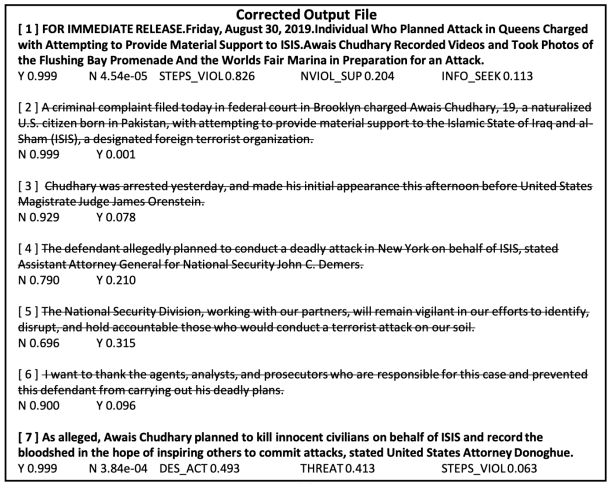
concerns in the training dataset and then provide us a more robust training set as part of the HITL process. This iterative effort recognizes that our training dataset is imbalanced by labels that overfit some of the labels during training. Moreover, we observe that model has degraded precision due to false positives (mostly in the form of classifying sentences for radicalization indicators where there is none present) and therefore we implement a two-phased processing pipeline as shown in Figure 5.4.

The screening model itself needed some of its own distinct training dataset that includes all the sentences that were manually labeled as having an indicator present and over 1000 sentences gathered during model testing that was deemed not relevant to radicalization. To illustrate the two-phased pipeline, Figure 5.5 shows 7 sentences of a Department of Justice public affairs statement that announced charges on a suspected radicalized individual (DOJ Public Release, 2021).

Figure 5.5a shows the result of the indicator classification model of the sentences without applying the screening. Therefore, the model returns classifications for sentences that are not relevant to radicalization behaviors (e.g., some statements made to make the public aware and describe the conduct of the investigation or the prosecution process). Figure 5.5b depicts the results of the screening model ('Y' means relevant to radicalization behaviors, and 'N' otherwise). The screening model removes from consideration 5 sentences an analyst or coder would not need to consider when attributing behaviors to a person of interest and leaves only 2 sentences that provide essential information. Therefore, the screening model improves the classification results to a certain level. However, the imbalanced and small training dataset mitigates the performance of the classification significanlty. Therefore, we further investigate other approaches to improve our classification tasks



(a) Output file of sample sentences scored using the multi-label classification model without screening



(b) A corrected output file from the two stage pipeline when the screening model filters for relevance, and then returns only the classification scores for those sentences which provide radicalization indicator information

Figure 5.5: Comparison of multi-label text classification with a screening model

and we identify pre-trained NLP models can be useful to improve the classification with their prior knowledge.

5.5.1 Transfer learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. Transfer Learning differs from traditional machine learning in that it is the use of pre-trained models that have been used for another task to jump start the development process on a new task or problem (*A Gentle Introduction to Transfer Learning for Deep Learning*, n.d.). Transfer learning has been gained a significant advancement on image classification. `Keras` [7], a Python machine learning library provides many pre-trained deep learning models that are made available alongside pre-trained weights for image classification (*Keras Applications*, n.d.).

To inspect the transfer learning capability for text classification, we have used a model called BERT (Devlin, Chang, Lee, & Toutanova, 2018), which stands for bidirectional encoder represen-

[7] https://keras.io/

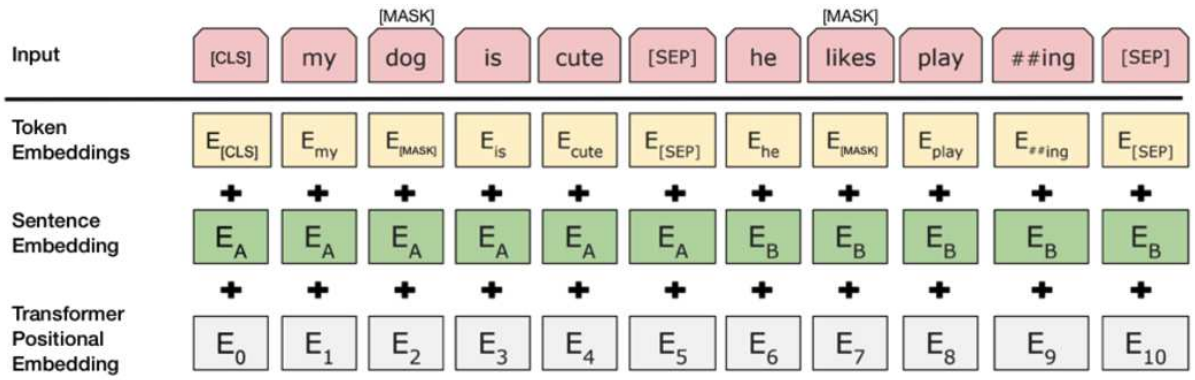


Figure 5.6: BERT input representation

tations from transformers. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. There are two steps in the BERT framework: 1) pre-training and 2) fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. BERT advances the state of the art for eleven NLP tasks. BERT is a generic NLP library which allows to customize the last layer of neural network to achieve various NLP tasks. Fundamentally, BERT excels at handling what might be described as 'context-heavy' language problems. BERT's input representation depicts in Figure 5.6.

5.5.2 BERT experiments and results

Figure 5.7 shows a training dataset we utilize for multi-label text classification that verifies the imbalance of the data by labels. We use the Stratified kfold cross-validation technique to address this concern, which is an extension for regular kfold cross-validation. Rather than the splits being completely random, the ratio between the target classes is the same in each fold in the entire dataset. In that way, we cover the total dataset as shown in Figure 5.8 and mitigate the overfits and irregularities of the dataset. In our work, we use k=10 because many studies show that

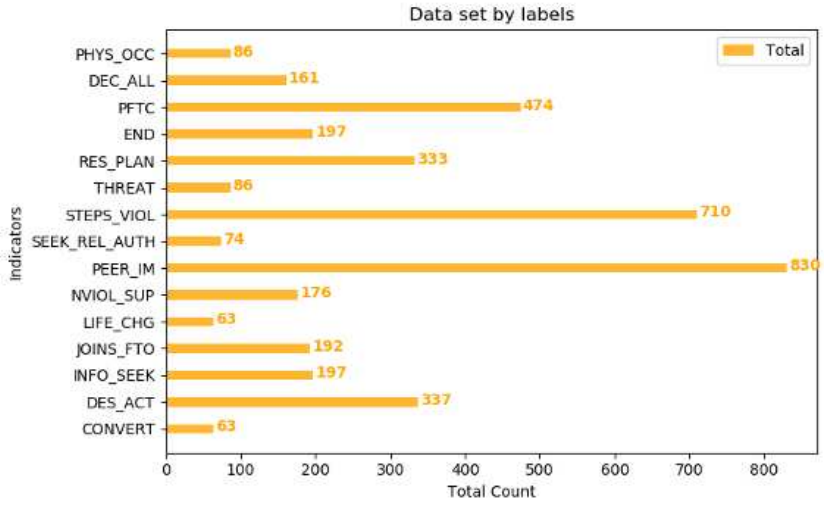
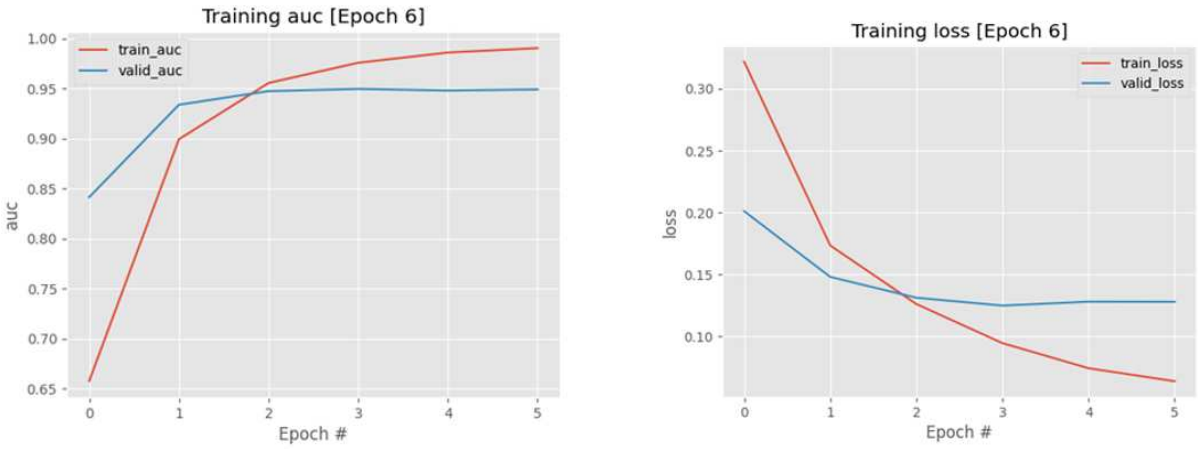


Figure 5.7: A multi-label dataset by labels

Figure 5.8: K-fold cross-validation when k=5

k=10 provides a reasonable trade-off of low computational cost and low bias in estimating model performance (Machine-learning Mastery, 2021). We further evaluate the classification accuracy with different k value to find optimum k value. Therefore, we generate 10 different train-test datasets for train 10 classification models. The results shown in the following are on average, among 10 train-test splits.

Even though traditional machine learning classification models take a significant number of epochs to train the network, BERT can converge to the optimum model performance within a few epochs. Figure 5.9 depicts the training/validation accuracy and training/validation loss of each epoch and validates the model converges with the smaller number of epochs. In this experiment,



(a) Training and validation accuracy

(b) Training and validation loss

Figure 5.9: Graphs for training BERT with radicalization dataset

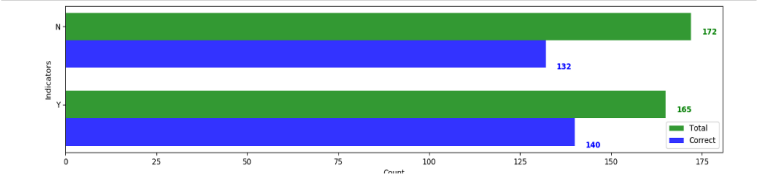


Figure 5.10: Multi-label text classification screening results.

too, we use our two stages multi-label classification as depicted in Figure 5.4. Figure 5.10 shows the results for the screening model that describes the relevance to the radicalization context. Green bars show the total test data set with 'Y' and 'N' labels, while blue bars interpret the correct classification counts. Figure 5.11 depicts the results for the complete classification model with 15 labels. The green and blue bars represent the total and correctly classified counts, respectively. Here, we consider a correct classification when only at least two labels are correctly classified among 3 labels attached to a text segment.

Even though our training data is imbalanced and small, the transfer learning approach via the BERT pre-trained NLP model assists in enhancing the classification accuracy to over 90%. At the same time, social scientists identify the deficiencies in the training datasets while annotating text segments, and we iteratively improve the robustness of the classification accuracy of the models. We further recognize the multi-label text classification is an ideal strategy amid other NLP approaches to this particular information extraction. With this work, we assist social scientists and

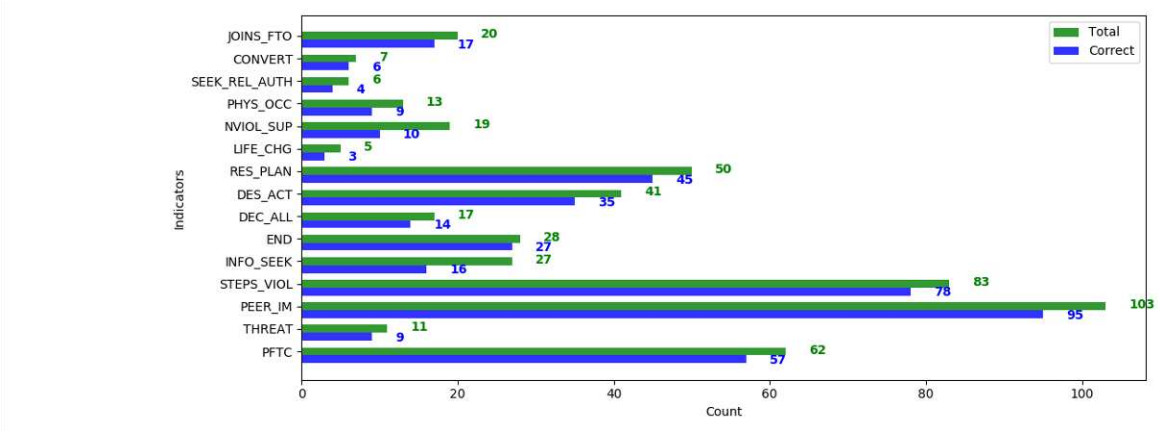


Figure 5.11: Multi-label text classification results based on each label.

coders in performing their data annotation process efficiently with more accuracy while building a rich information network about extremists and suspicious individuals. That leads us to the next problem of storing high-connected and dynamic datasets for long-term for further evaluations and studies. Chapter 6 discusses the data storage and querying the data for identifying threat advancements and suspicious individuals and groups.

Chapter 6

Enhancing Investigative Graph Search with Graph Databases

6.1 Introduction

The growth of social networking has resulted in the generation of data that can be used to identify terrorist activities, mental health issues, and consumer behavior. Such data is dynamic in nature whose analysis can reveal hidden or emergent behavior. Such data can be represented as dynamic graphs (Carley et al., 2014; Klausen, Libretti, et al., 2018), and often observable in interactions via social networks (Klausen, Marks, & Zaman, 2018; Lara-Cabrera et al., 2017). Organizations often try to detect insider threats using performance-related and technical indicators recorded over time (CERT Division Software Engineering Institute, 2019). Businesses track an individual's online purchase habits to determine the potential for future purchases (Edelman & Singer, 2019, 2015). Identification of behaviors that precede suicide is of vital interest (Jashinsky et al., 2014; Olson, 2011) to mental health professionals. Graph-based models that allow mining and tracking of such behavior appears to be a promising approach.

We are particularly interested in identifying homegrown violent extremism, which is a cause of concern in several countries (Klausen et al., 2016). A person gets radicalized through a sequence of activities; these include the consumption of extremist ideas and propaganda through internet sources and the association with other radicalized peer groups (King & Taylor, 2011; Klausen et al., 2016; Klausen, Libretti, et al., 2018). Sometimes preparatory tasks and/or attacks have been carried out as groups. It is a daunting challenge to detect how indicators are emerging among individuals or groups in a large population.

This work addresses large-scale network mining and analytics essential to extract and track partial and emergent profile matches of individuals and groups. An inexact graph matching tech-

nique is presented together with results for three investigative domains: radicalization, patients, and crime pattern detection. We discuss the computational complexities of the proposed approach. Further, we depict the importance of graph databases as a data store while utilizing database features like caching for investigative data analysis.

6.1.1 Datasets

Three data sets are used to validate and evaluate the proposed algorithms.

Radicalization dataset (RD) - This data, synthetically generated, mimics the behavior of homegrown radicalized extremists (Klausen, Libretti, et al., 2018; Klausen et al., 2016). It is used to evaluate the query performance and the scalability. The sizes of different datasets are depicted in Table 6.5. We also show the effect of the in-memory database cache on the query performance. A query graph is specified that represents the generic behavior profile of an extremist, criminal or their group. The query graph represents all potential behaviors of individuals of interest, and the score mechanism calculates the similarity between each individual or group and the query graph.

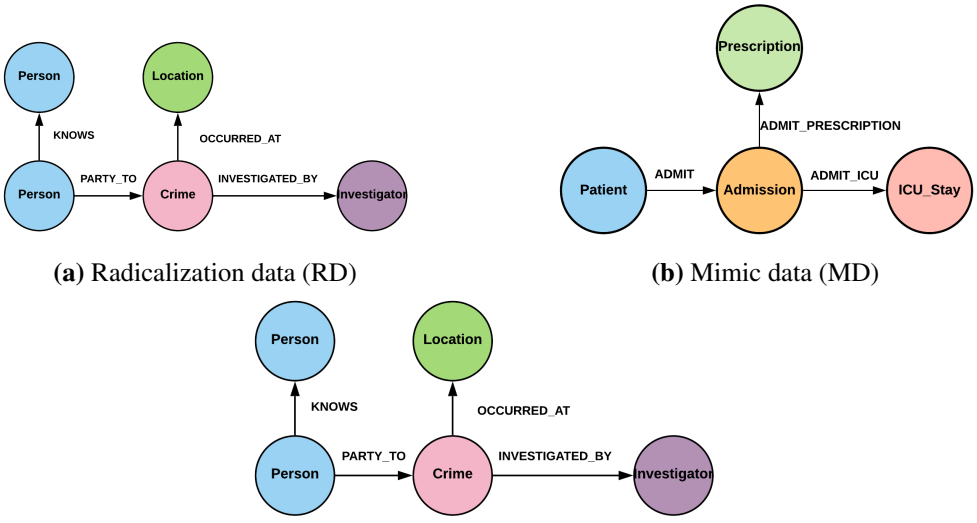
Mimic Dataset (MD) - We use MIMIC III medical dataset (Johnson et al., 2016) that is a large, freely-available database of clinical visit records of Intensive Care Unit (ICU) patients between 2001 and 2012. The information of 1000 patients was extracted and produced 84687 nodes and 83686 edges. The algorithm was applied to find similar patterns of admitted patients in a specific ICU with certain prescriptions.

Crime dataset (CD) - A data set that can be accessed from the Neo4j sandbox (Neo4j.com, 2020a) containing the data related to street crimes in Greater Manchester, UK. The dataset was extracted from public sources about the prevalence, location, and type of crime (Government Digital Service UK, 2020), but it does not include any real information about persons related to the crimes. The dataset has been enriched to include fake criminals, investigators, and timestamps for demonstration purposes. The dataset includes 369 criminals, and consists of 61521 nodes and 105840 edges. This dataset was used to search for similar crime patterns with respect to a criminal or a specific location.

6.2 Description of investigative data

Social network data are inherently heterogeneous, connected, and labeled. Mining such comprehensive social and knowledge networks requires a wide range of analytical methods. While our proposed method and library is applicable for a wide range of applications, here we apply it to radicalization, patient's ICU stays, and crime data, which are contemporary investigative domains. All the data sets are directional and include node labels, properties, and relationship types, as illustrated in Figure 6.1 which depicts the data schema for the three cases. We use only the subset of node types in Figure 6.1b for patients data analysis and in Figure 6.1c for crime data analysis. Although an expansive analysis of patients' data may use additional types such as Drug code, Services, and Lab items, crime data may utilize other types such as Vehicle, Object, and Phone.

When a search criterion is defined, a *query focus* node is included to point to the starting node of the search. In the radicalization and mimic data sets, the *query focus* is the 'Person' node and 'Patient' node label respectively. In the crime dataset, the 'Location' or the 'Person' labels may be used for location-based and criminal-based analysis respectively. Our procedures allow to specifically indicate the significance of the highly important activities/indications as *redflags*. The *redflag* value prioritizes specific activities while appending an extra weight to the similarity measure calculation. For radicalization data, the node categorization is done on the basis of data schema and the characterization of radicalization indicators (B. W. Hung & Jayasumana, 2016). In the radicalization dataset, the 'Indicator' node holds a behavioral indicator type as a property ('name'). The *redflag* indicator types ('RF'), ordinary indicator types ('IND'), and other node types are depicted in Table 6.1. In the mimic dataset, the 'Admission' node contains the details of the hospital admission. In this dataset, we do not consider a *redflag* node and all the node types are discussed in Table 6.2. In the crime dataset, the 'Crime' node holds a crime type as a property ('type'). The *redflag* crime types ('RF'), ordinary crime types ('IND'), and other node types are depicted in Table 6.3.



(a) Radicalization data (RD)

(b) Mimic data (MD)

(c) Crime data (CD)

Figure 6.1: Data schema of the datasets

6.3 Radicalization, mimic, and crime data schema

The schema corresponding to the radicalization data in Figure 6.1a represents an example of the preliminary behavior of a radicalized person. 'Person' is an individual who connects with others through the *KNOWS* relation. This dataset includes a maximum of six different behavioral indicators. An indicator is a suspicious activity/behavior that connects with persons through *EXHIBITS* relation. A radicalized individual may have multiple Social Media Accounts ('SM_Acc'). Such account links are marked with *HAS* relationship and corresponding posts are marked with *POSTS* relationship.

In the mimic data schema shown in Figure 6.1b, 'Patient' connects with an 'Admission' node via *ADMIT* relationship. An admission node links to a ICU stay node via *ADMIT_ICU* relationship. An admission node also connects to a prescription node using *ADMIT_PRESCRIPTION* relationship.

In the crime data schema depicted in Figure 6.1c, 'Person' denotes a person such as a criminal in this context, and connects to other nodes using different types of relations. In this work, we indicate all person-to-person relationships using *KNOWS* relationship. *PARTY_TO* relationships link criminals to their crimes. A crime node has a location that connects to the node with *OC-*

Table 6.1: Node types in radicalization dataset (RD). 'Indicator' node was divided into two node types as RF (Red flag), and IND (Indicator) based on the indicator type.

Node type	Node name	Indicator type	Description
QF (Query Focus)	Person		Initial node represents the subject where the search starts
RF (Red Flag)	Indicator	• Carried out an attack • Detonated a bomb	A high-risk behavior that needs immediate further investigation
IND (Indicator)	Indicator	• Purchased weapons • Received training • Referred to radicalized materials • Suspicious travel	A behavior that should be traced further
SM	Social Media Account		Represents a social media account of a Person
EP	Extremist Post		Represents a post containing radicalization related content.

Table 6.2: Node types in mimic dataset (MD).

Node type	Node name	Admission/ICU types	Description
QF (Query Focus)	Patient		The initial node to start patient anlaysis
IND (Indicator)	Admission	• Emergency • Elective • Urgent • Newborn	Details of a patient's hospital admission
ICU_Stay	ICU stay	• MICU (Medical ICU) • CCU (Coronary CU) • NICU (Neonatal ICU) • TSICU (Trauma surgery ICU) ...	Details of ICU stay including the duration and the type of the ICU
Presciption	Presciption		Contains medication related order entries

Table 6.3: Node types in crime dataset (CD). 'Crime' node was divided into two node types as RF (Red flag), and IND (Indicator) based on the crime type.

Node type	Node name	Crime type	Description
QF (Query Focus)	Person		The initial node to start search for criminal-based analysis
QF (Query Focus)	Location		The initial node to start search for location-based analysis
RF (Red flag)	Crime	• Possession of weapons • Criminal damage and arson • Violence and sexual offenses • Drugs • Burglary	High-risk crimes that law enforcement need to prioritize for taking actions straight away
IND (Indicator)	Crime	• Public order • Theft from the person • Other theft • Vehicle crime • Robbery • Other crime • Shoplifting • Bicycle theft	Comparatively less severe crimes than RF, but still need further investigations
INV	Investigator		Represents a law-enforcement personnel who conducts the investigation

CURRED_AT relationship. A crime may also link to an investigator (generally a police officer) via *INVESTIGATED_BY* relationship.

6.4 Evolution of graph databases

Graph databases are beneficial for pattern-based querying over large volumes of data that are characteristic of our problem domain, where both the data and the relationships between the data play a crucial role. Unlike SQL or NoSQL databases, they are designed to treat the relationships between data as equally important as the data itself (Neo4j.com, 2020c). The nodes and edges are considered as separate data structures in the database. In graph data modeling, data is

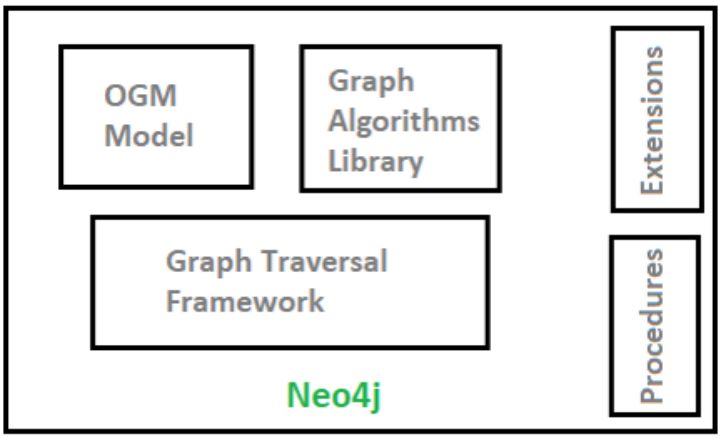


Figure 6.2: High-level architecture of Neo4j graph database

organized as nodes, relationships, and properties. Graph databases reshape graph problems and facilitate novel approaches. Neo4j (Neo4j.com, 2019b), the most widely used graph database, includes basic graph traversal algorithms and allows for user-defined procedures. The developer libraries simplify the access of graph data structures and allow for the implementation of custom query procedures that addresses unique requirements. Generally, we represent a graph through an adjacency matrix, but graph databases, especially Neo4j provides flexibility to think about graph problems in an object-oriented manner. The built-in library of graph algorithms optimized for the database (Needham & Hodler, 2019; Neo4j.com, 2019a) including those on centrality, community detection, and pathfinding, facilitate analytic insights for investigators. Neo4j is implemented based on Java object-oriented programming language, and nodes and edges of a graph represent objects through its Object Graph Mapping (OGM) model. Neo4j also supports various other languages via APIs such as python, PHP, javascript. Figure 6.2 interprets key functionalities of the Neo4j database. It has an Object Graph Matching (OGM) Model, which matches the nodes and edges to objects. As we mentioned, it allows reforming graph problems in an object-oriented manner. Graph Algorithm Library and Graph Traversal Framework consists of basic graph algorithms such as depth-first search and breadth-first search. Another key feature is that Neo4j facilitates implementing custom procedures and extensions based on the user requirement.

In this research, we used a Neo4j graph database as the data storage for heterogeneous knowledge networks. We implemented an algorithm that is based on sub-graph pattern matching to fetch profiles of interest from the graph database. The algorithm is based on sub-graph pattern matching using a scoring mechanism. The algorithm is implemented as a set of Neo4j procedures called PINGS (Procedures for INvestigative Graph Search) which facilitates custom queries. We are focusing on the sub-graph isomorphism problem, which is NP-hard (Asiler & Yazıcı, 2017). This work focuses on the inexact sub-graph isomorphism problem for investigative pattern detection. We search a large data graph for instances of a pre-defined query graph while allowing for inexact matches.

6.5 Investigative search using graph databases

We discuss the terms and notations before describing the proposed algorithm.

Definition 1. Investigative graph

An investigative graph is a directed graph $G = (V[L, P], E[R])$, where

- V is a finite set of nodes, with cardinality n, i.e., $n = |V|$
- E is a set of edges, $E \subseteq V \times V$, where (v, v') is an edge from v to v'
- L is the set of labels for node types, e.g., $L = \{Person, Indicator, Crime, Post, l\_query, ...\}$
- P is the set of properties/attributes associated with a node
- R is a set of labels for edge types, e.g., $R = \{exhibits, knows, posts, ...\}$

Data graph is denoted by $G = (V_G[L_G, P_G], E_G[R_G])$, and a query graph is denoted by $Q = (V_Q[L_Q, P_Q], E_Q[R_Q])$.

Definition 2. Node match

A node based on the requirement stated in Q and the features of a graph database can be matched in two ways: some nodes are matched on the basis of labels and others on the basis of properties.

- *Label match*

  Nodes $u \in V_Q$ and $v \in V_G$ have a label match, if $l'(u) = l(v)$ where $l' \in L_Q$ and $l \in L_G$.

- *Property match*

  Nodes $u \in V_Q$ and $v \in V_G$ have a properties match, if $p'(u) = p(v)$ where $p \in P_Q$, $p' \in P_G$.

Definition 3. Edge match

Two edges match if they both agree on the direction and relationship as stated below.
Edges $e_u = (u, u') \in Q$ and $e_v = (v, v') \in G$ match, if $r'(e_u) = r(e_v)$ where $r' \in R_Q$, $r \in R_G$.

We develop Neo4j procedures to implement similarity measures in order to detect individuals (*individualSimilarity*) and groups (*neighborhoodSimilarity*). The *implicitIndividualSimilarity* procedure is introduced for specific cases where the query graph is dynamic and needs frequent changes. This procedure allows defining only a *queryFocus* node to implicitly find the respective query graph and search for similar patterns. So, it facilitates evaluating diverse query patterns without changing the core algorithm in the *individualSimilarity* procedure. In these custom Neo4j procedures, the query graph is stored within the database with a different node label. Almost all the inexact subgraph isomorphism problems can have the input parameters proposed in procedures. e.g., *similarity threshold* is to define the inexactness of the query while *query label* and *query focus label* specify the query graph. *red flag* is an optional to bias a node type as required.

Figure 6.3 shows an example query graph for the radicalization dataset. A complete match with the query graph indicates potential radicalization. It signifies all the indicators one could expect from a violent extremist. Querying for this pattern corresponds to searching for individuals who have exhibited some or all of these indicators, using which we rank high-risk individuals. We calculate similarity scores for different subgraphs in the database with respect to the query graph and fetch matching subgraphs whose similarity scores exceed a threshold value.

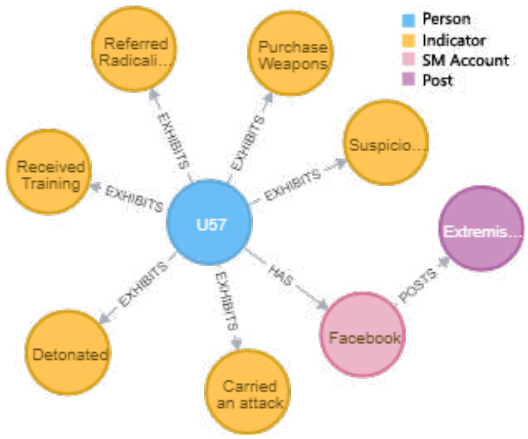


Figure 6.3: An example query graph for radicalization dataset (RD); six different behavioral indicators (orange) are included.

6.5.1 Individual similarity measure procedure

The Neo4j procedure consists of these 4 input parameters:

$$individualSimilarity(similarityThreshold, \\ redFlagMultiple, queryLabel, queryFocusLabel) \tag{6.1}$$

- *similarityThreshold* – A similarity score that is normalized to (0,1) is calculated for each user based on his activities. *similarityThreshold* score of the query graph is used to identify matching subgraphs.

- *redFlagMultiple* – This attribute is used to highlight the highest risk activities/indicators (B. W. Hung & Jayasumana, 2016). The *redFlagMultiple* (≥ 1) is used to multiply (weight) high-risk activities as a weighted (multiplicative) value. The end result is a prioritization of sub graphs by similarity score due to the presence of high-risk activities.

- *queryLabel* – A node in a Neo4j graph database can have multiple labels. This allows us to identify a query graph within the database by appending another label (eg: *'l_query'*). This allows the query graphs to be easily updated for different experiments.

- *queryFocusLabel* – The node label indicates the starting point of the algorithm. In our case, we provide the label of the person, *'Person'*. The algorithm then evaluates the similarity measure for each person.

For example, the *individualSimilarity* procedure call for an inexact similarity measure can be specified as follows. *individualSimilarity(0.7, 2, 'l_query', 'Person')*

Algorithm 1: *individualSimilarity*

inputs : Q : Query label
F : Query focus (QF) label
S : Similarity threshold
R : Red-flag multiple
output: G_M : Set of matched graphs

$G_M \leftarrow \emptyset$
$C \leftarrow getConfigurationList$
$G_Q \leftarrow getQueryGraph(Q, F)$
$S_f \leftarrow \{s' \mid s' \in \mathcal{V}_G \,\&\, matchNode(F, s')\}$
foreach $s' \in \mathcal{S}_f$ **do**
    $g_m \leftarrow searchSimilarGraphs(s', G_Q, R, C)$
    $s_m \leftarrow getMatchedScore(g_m)$
    if $s_m \geqslant S$ **then**
        Add g_m to G_M
return G_M;

6.5.2 Neighborhood similarity measure procedure

Sometimes radicalized individuals work with individuals or groups with similar interests for specific tasks. Calculating individual similarity scores for single-person subgraphs is inadequate to detect such collectively suspicious behavior as the indicators are effectively dispersed among the group members (B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019). We propose a neighborhood measure in order to identify such group activities.

$$neighborhoodSimilarity(similarityThreshold, redFlagMultiple, queryLabel, queryFocusLabel) \tag{6.2}$$

Algorithm 2: *searchSimilarGraphs*

inputs : s' : QF node in V_G
Q : Query graph
R : Red-flag multiple
C : Configuration list
output: g_m : Matched graph

```
1  g_m ← ∅, D_M ← s'
2  foreach q' ∈ Q do
3      M ← ∅
4      foreach g' ∈ D_M do
5          if matchNode(g', q', W, C, g_m) then
6              Remove g' from D_M
7              R_Q = {r_q | r_q ∈ Q_E & directedEdges(q')}
8              foreach r_q ∈ R_Q do
9                  R_G = {r_g | r_g ∈ G_E & directedEdges(g')}
10                 foreach r_g ∈ R_G do
11                     m = matchEdge(r_q, r_g, g_m)
12                     if m != null then
13                         M ← m
14         D_M ← M
15 return g_m;
```

The procedure uses the same input parameters as the *individualSimilarity* and invokes Algorithms 3 and 4 to detect suspicious groups.

6.5.3 Implicit individual measure procedure

In practice, there are specific cases where query graphs need to be adaptively and dynamically modified for complex analysis. In such cases, rather than explicitly defining a query graph via a label (eg:'l_query'), only a *query focus* node is defined and is able to implicitly find the respective query graph and follow the same similarity measure functionality. In crime analysis, the query graph needs to be changed frequently in both criminal-based and location-based analysis. Thus, the implicit similarity measure is used for crime analysis. The procedure parameters are changed so that crime patterns associated with either a criminal or a location can be searched. We do this by defining a *query focus* node. The modified procedure definition appears below.

$$\begin{aligned} &implicitIndividualSimilarity(entity, identifier, identifierValue, \\ &relationshipToFocus, redFlagMultiple, similarityThreshold) \end{aligned} \tag{6.3}$$

Crime pattern analysis offer more flexibility when compared to radicalization detection using canonical violent extremist patterns as it allows analysts to select or define a particular crime pattern within the data as a query graph. The procedure searches for the existence of similar patterns (by criminal or location) throughout the database to identify others perpetrating similar crimes or locations that witness them. The first three inputs define the root node for analysis. *Entity* represents the node label (in this dataset the *Location* or the *Person*). *Identifier* is the property of the node to be recognized and *identifierValue* depicts the particular property value. It is also essential to provide the relationship type to the crime node represented with the parameter *relationshipToFocus*. *redFlagMultiple* highlights high-risk crimes based on Table 6.3. *similarityThreshold* serves the same function as before. Criminal-based pattern detection invocation example: *implicitIndividualSimilarity('Person','nhs_no','444-91-2379', 'PARTY_TO', 1, 0.8)*

```
{
  "nodes": [
    {
      "label": "Person",
      "attrs": []
    },
    {
      "label": "Indicator",
      "attrs": ["name"]
    }
  ],
  "red_flag": {
  },
  "neighbour_relationship_type": "KNOWS",
  "activity_node_type": "Indicator",
  "identifier_type": ""
}
```


Figure 6.4: An example configuration list (C)

Crime location-based pattern detection invocation example: *implicitIndividualSimilarity('Location', 'postcode', 'M5 3WT', 'OCCURRED_AT', 1, 0.9)*

Table 6.4: Important variable descriptions of algorithm 1, 2, 3, & 4

Variable	Description
G_M	A set of all matched graphs
C	Configuration list
G_Q	Query graph
S_f	A set of all query focus nodes in a data graph
s'	A query focus node in a data graph
g_m	Matched graph for a given s'
s_m	Similarity score for a given g_m
n_n	Neighbor query focus nodes for a given s'
n_m	Matched neighbor graphs for a given g_m

6.5.4 Individual measure algorithm

Algorithm 1 describes how to find the similarity measure. For better readability of pseudo code in algorithms, Table 6.4 depicts the descriptions of essential variables. The algorithm first finds all root user nodes (line 4). Then for each user node, it calls the *searchSimilarGraphs* function

(line 6). Neo4j graph database allows adding multiple labels or properties to any node or edge. So, we maintain a configuration list (C) shown in Fig 6.4, which is unique to a particular data schema that includes labels and properties of nodes which need to be matched. With the key-value map structure, the time-complexity of traversing through the C is reduced from $O(l_c)$ to $O(1)$ where l_c is the size of C. *searchSimilarGraphs* function calculates the similarity score while searching for a query graph in a data graph. If the calculated similarity score is larger than or equal to a given *similarityThreshold*, that data graph is added to the result set (M).

Algorithm 2 illustrates the process of retrieving similar sub-graphs based on a query focus node s'. The traversal initiates from the nodes in the query graph (line 2) then match the data graph's nodes. The graph matching consists of *matchNode* (**Definition 2**) in line 5 and *matchEdge* (**Definition 3**) in line 11 to match nodes and edges respectively. With an edge match, it signifies that the corresponding end node also matches. Further, the metadata in the configuration list (C) is utilized to obtain the matching criteria. The algorithm keeps a *queue* data structure to collect possible matched data nodes. As explained, the traversal launches based on the query graph (line 2), which is comparatively smaller and avoids redundant traversal in the data graph. An example execution of the individual similarity for a given query graph and a *query focus* node is depicted in Figure 6.5.

6.5.5 Neighborhood measure algorithm

Algorithm 3 explains the proposed neighborhood measure. Lines 1-6 are similar to those in the *similarityMeasure* algorithm and the matched graph of the primary *query focus* node is retrieved. In line 7, the neighbor nodes related to the primary *query focus* node are fetched. Then, Algorithm 4 is called to retrieve the matched graphs of neighbors. Algorithm 4 also introduces an aggregating schema to measure collective indicators exhibited by a person and his associates. Then it performs the *searchSimilarGraphs* function (Algorithm 2) for each neighbor to get matched graphs (line 5). Each neighbor should contribute at least one unique indicator compared to the primary *query focus* node for it to be considered as a contributor to the group. If a neighbor shares an indicator with

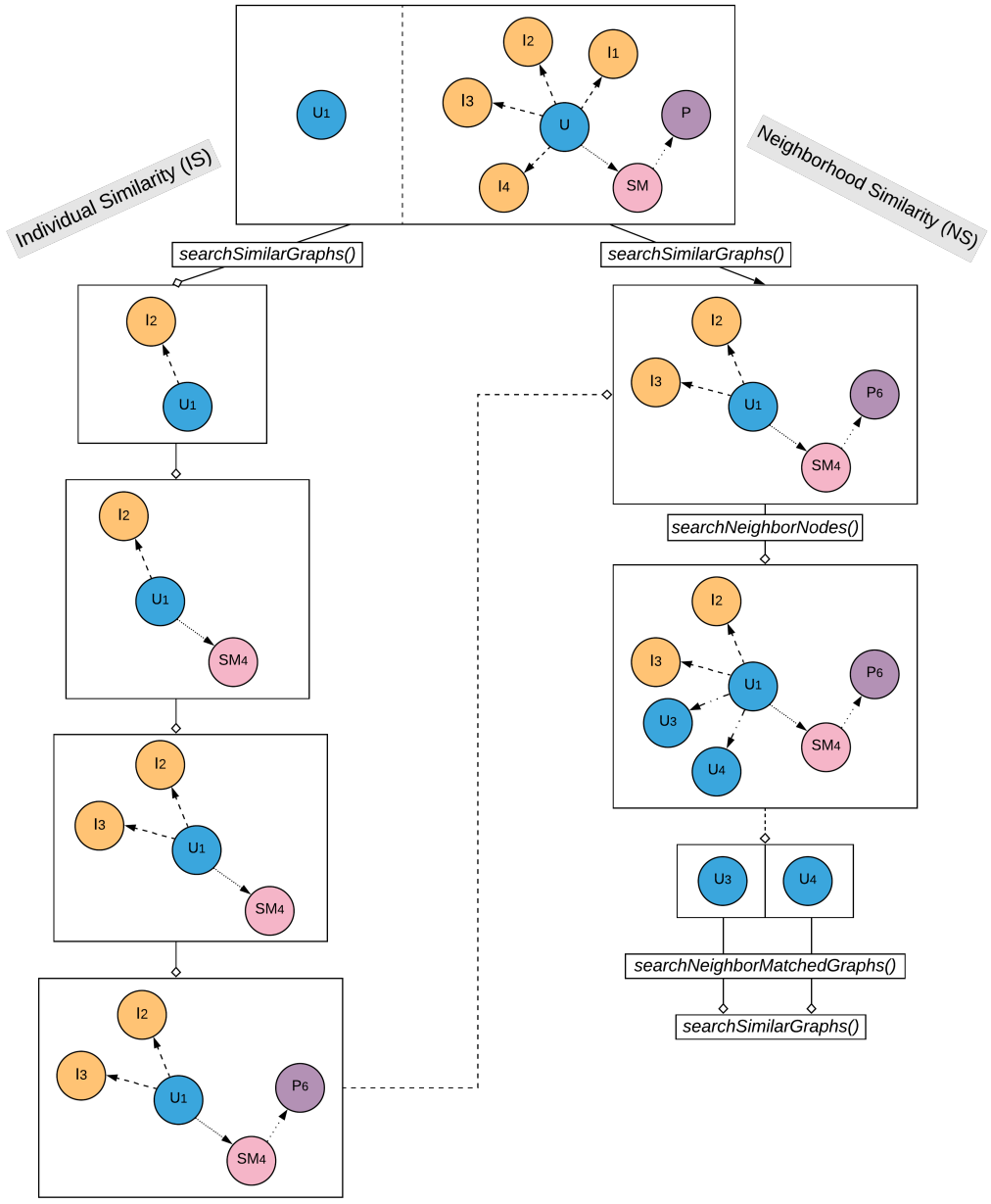


Figure 6.5: A sample execution of the individual similarity (IS) and the neighborhood similarity (NS). Top level represents a node in the data graph [left] and a query graph [right]. The left-side of the chart depicts the flow of individual similarity (IS) and the right-side depicts the flow of neighborhood similarity (NS). In individual similarity, it invokes *searchSimilarGraphs()* function for 'U_1' *query focus* node and discovers I_2, SM_4, I_3, and P_6 matching nodes respectively. In neighborhood similarity, *searchSimilarGraphs()* function is invoked initially to fetch the IS matching graph. Then, *searchNeighborNodes()* function searches for all neighbor nodes of 'U_1'. In this case, 'U_3' and 'U_4' *query focus* neighbor nodes are retrieved. Then, *searchNeighborMatchedGraphs()* function is called to execute *searchSimilarGraphs()* function for each *query focus* neighbor node while updating collective similarity score.

the primary node, it is also considered but no value is added to the similarity measure. In this way Algorithm 3 checks whether each neighbor could be a potential contributor to achieve a set of indicators in the query graph. The function *updateCollectives* maintains collective indicators of a group which is later matched with the query graph. The *checkEligibility* function checks the neighbor's eligibility of being a contributor to the group. Figure 6.5 illustrates an example execution of the neighborhood similarity for a given *query focus* node and a query graph.

Algorithm 3: *neighborhoodSimilarity*

inputs : Q : Query label
F : Query focus (QF) label
S : Similarity threshold
R : Red-flag multiple
output: G_M : Set of matched graphs

```
1  G_M ← ∅
2  C ← getConfigurationList
3  G_Q ← getQueryGraph(Q, F)
4  S_f ← {s' | s' ∈ V_G & matchNode(F, s')}
5  foreach s' ∈ S_f do
6      g_m ← searchSimilarGraphs(s', G_Q, R, C)
7      n_n ← searchNeighborNodes(g_m)
8      n_m ← searchNeighborMatchGraphs(n_n, S, G_Q, R, C, g_m)
9      G_M ← n_m
10 return G_M;
```

6.6 Computational complexity of individual and neighborhood measure algorithms

We evaluate the time and space complexity of *individual measure* and *neighborhood measure* for worst-case scenarios. In the analysis, we denote the number of nodes (vertices) in query graph Q by $|V_Q|$, number of nodes (vertices) in data graph G by $|V_G|$. We also use $\mathbb{D}_Q^q$ and $\mathbb{D}_G^q$ to denote the maximum node degree in the query graph and the maximum node degree in data graph, respectively, where q denotes the node degree related to the query relationship types with the

exception of the neighbor (person-to-person) relationships. $\mathbb{D}_G^n$ denotes the maximum neighbor node degree in the data graph and the number of *queryFocus* nodes (vertices) in the data graph, which is denoted $|V_G^{qf}|$.

6.6.1 Time complexity

Algorithm 4: *searchNeighborMatchedGraphs*

inputs : n_n : Set of Neighbor nodes
G_Q : Query Graph
R : Red-flag multiple
C : Configuration List
g_m : Matched Graph
S : Similarity Threshold
output: N_G : Set of matched graphs

```
1  initialCSet ← updateCollectives(G_Q, g_m)
2  activityCSet ← ∅
3  N_G ← ∅
4  foreach n'_f ∈ n_n do
5      g_m ← searchSimilarGraphs(n'_f, G_Q, R, C)
6      nodeCSet ← updateCollectives(G_Q, g_m)
7      if checkEligibility(initialCSet, nodeCSet) then
8          activityCSet ← applyCollectives(activityCSet, nodeCSet)
9          N_G ← g_m
10 if checkSimilairtyScore(activityCSet) ⩾ S then
11     return N_G;
```

First, we focus on the *matchNode* function. Based on the *matchNode* definition (**Definition 2**), it has a label plus property match where a label and one or a few properties are considered to be matched. Therefore, *matchNode* function is considered to have $O(1)$ worst-case time complexity. As explained in Subsection 6.5.4, the *matchEdge* function includes the *matchNode* to match not only the relationship type but also the next node. The relationship type check is a onetime check that has $O(1)$ worst-case time complexity. Therefore, the worst-case time complexity of the *matchEdge* function is $O(1)$. As a graph database related similarity measure, the query graph also is stored inside the database with a different node label (Q_l). In Algorithm 1, the query graph is

fetched initially. Since, the label information is implicitly available in the Neo4j database, filtering operation with respect to node labels can be considered as $O(1)$ (Asiler & Yazıcı, 2017). Then, we fetch all the *queryFocus* nodes, the worst-case time complexity of filtering all *queryFocus* nodes is taken as $O(1)$ with the same performance in a graph database. Then, the *searchSimilarityGraphs* (Algorithm 2) is invoked for each *queryFocus* label. It traverses through all the query nodes and maximally D_M will take $|V_Q|$ number of nodes to traverse in the exact match case. The query and data relationship type match is an iterative process that gives the worst-case time complexity $O(\mathbb{D}_Q^q \times \mathbb{D}_G^q)$. Then, we can claim the worst-case complexity of the similarity measure algorithm (explained in Algorithm 1 and 2) is $O(|V_G^{qf}| \times |V_Q| \times \mathbb{D}_Q^q \times \mathbb{D}_G^q)$.

The neighborhood measure explained in Subsection 6.5.5 searches for groups of individuals who collectively satisfy the query graph (Q). For a particular *queryFocus* node, it takes the neighbor(person-to-person) nodes and executes the *searchSimilarityGraphs* function (Algorithm 2) for each neighbor. Then, the array that tracks the collective exhibition of activities is updated, which can be considered as a linear operation. Therefore, the worst-case complexity of the neighborhood measure algorithm (explained in Algorithm 3 and 4) is $O(|V_G^{qf}| \times |V_Q| \times \mathbb{D}_Q^q \times \mathbb{D}_G^q \times \mathbb{D}_G^n)$.

6.6.2 Space complexity

The query graph (Q) is also formed inside the Neo4j database with an additional node label (eg: Query). In both similarity and neighborhood measures, the query graph is retrieved initially with $O(|V_Q|)$ auxiliary space complexity. The configuration list (C) is also fetched but can be considered as negligible space usage compared to the graph data storage. Then, all *queryFocus* nodes in the data graphs are fetched are stored with $O(|V_G^{qf}|)$ auxiliary space complexity. For the each *queryFocus* node, at most $|V_Q|$ (in the exact match case) nodes are stored in the *searchSimilarGraphs* function call. In each call, the values of the collective array are updated but size of the array remains a constant, which is linear, and exhibits negligible space complexity. So, we can claim the worst-case space complexity of the similarity measure is $O(|V_Q|(1 + |V_G^{qf}|))$. In the neighbor-

Table 6.5: Characteristics of radicalization datasets

Dataset	DS_1	DS_2	DS_3	DS_4
No of persons	100	1000	10000	100000
No of nodes	979	9627	96590	966572
No of edges	909	8178	78590	784053

hood measure, the matched nodes of neighbors are also stored additionally. In this case, the worst-case space complexity of the neighborhood measure can be claimed as $O(|V_Q|(1 + |V_G^{qf}| \times \mathbb{D}_G^n))$.

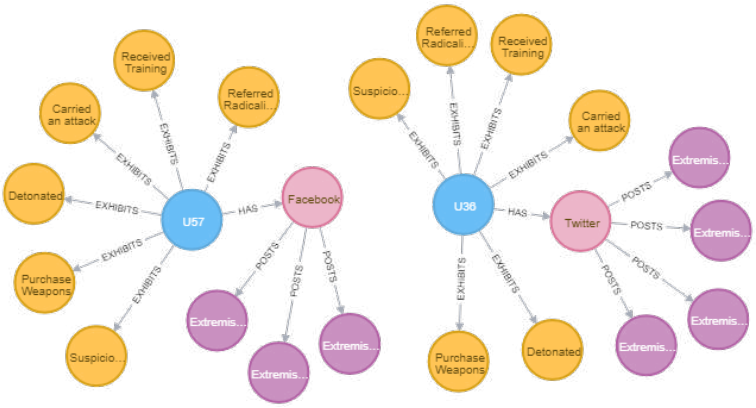
6.7 Experimental evaluation

We conducted similarity measure experiments for different graph database setups for *individualSimilarity*, *neighborhoodSimilarity* and *implicitIndividualSimilarity* procedures in PINGS library. We evaluated the query performance, including in-memory database cache impact, for different sized radicalization datasets.

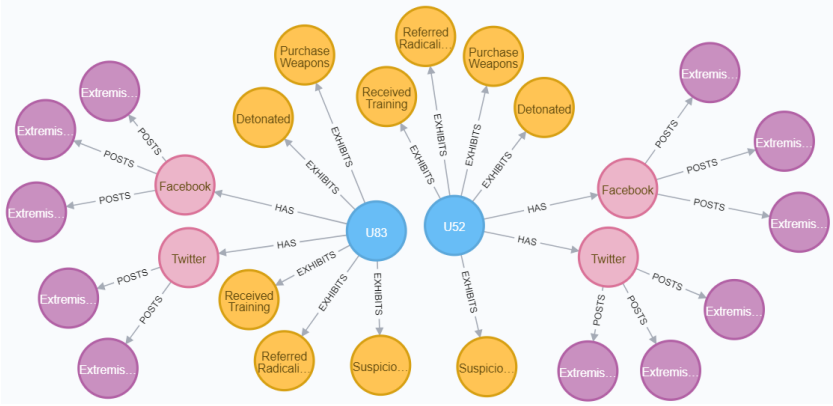
6.7.1 Analysis of radicalization data

Figure 6.6a shows the results for exact pattern match (*similarityThreshold=1* & *redFlagMultiple=1*) when executed on the query graph in Figure 6.3. As we explained in Section 6.5, the query graph is also defined inside the graph database using a distinct node label. In these custom Neo4j procedures, the query graph is stored within the database with a different node label. Figure 6.6a shows that users 'U57' (shown on the left) and 'U36' (shown on the right) are responses to exact matches for the query graph. This is an exact match on the prioritized indicators only. The other indicators, such as the number of social media posts or the number of social media accounts may differ.

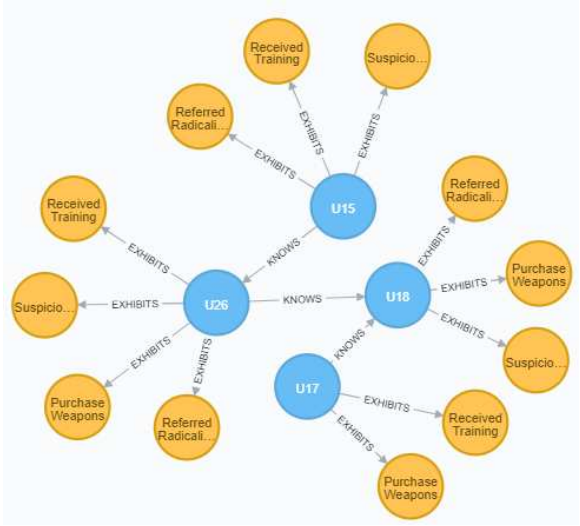
Examples of inexact match results are shown in Figure 6.6b when the *similarityThreshold* is 0.7 and *redFlagMultiple* is 1. Users 'U52' and 'U83' have demonstrated 5 (out of 6) indicators and they have used social media accounts to disseminate contents indicating or aimed at radicalization. Note that, we are able to detect look-alike suspicious behaviors which do not exactly match a given query graph.



(a) Individual measure for exact match

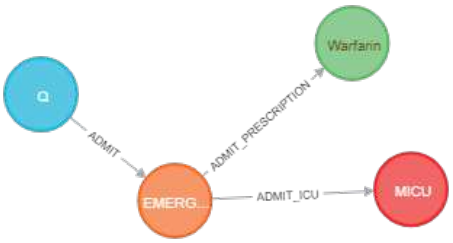


(b) Individual measure: Inexact match with similarity threshold 0.7

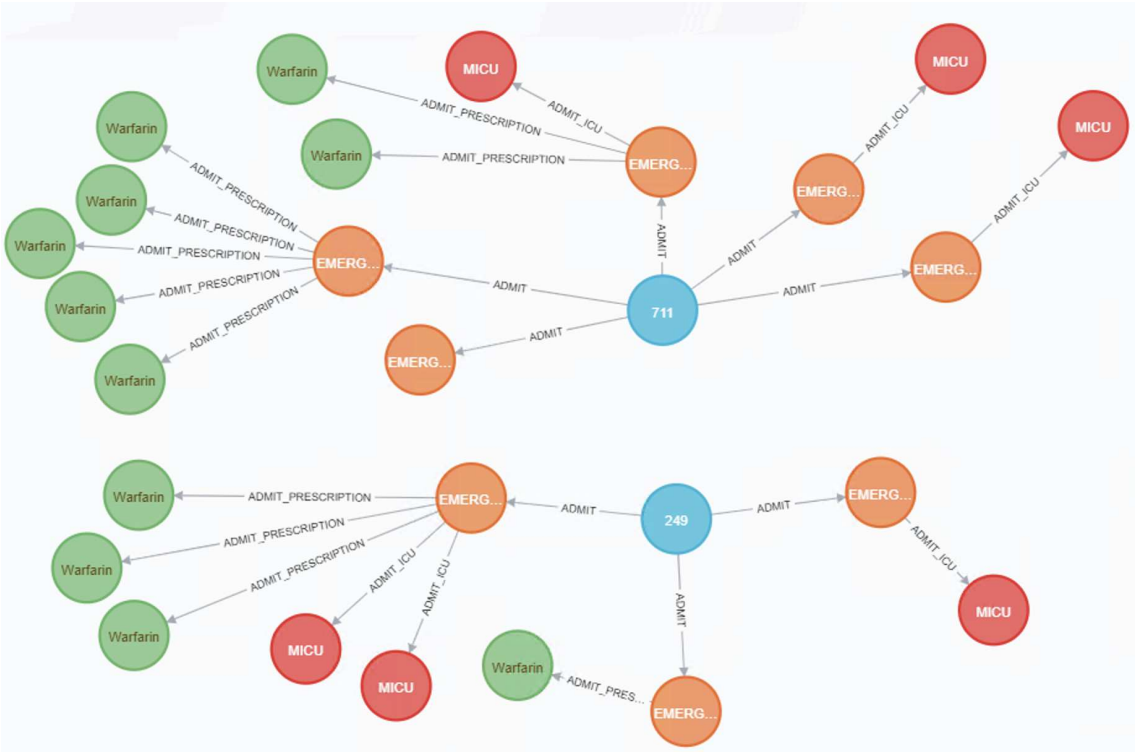


(c) Neighborhood measure: Inexact match with similarity threshold 0.8

Figure 6.6: Individual and neighborhood measure for Radicalization Data (RD)



(a) Query graph for a patient

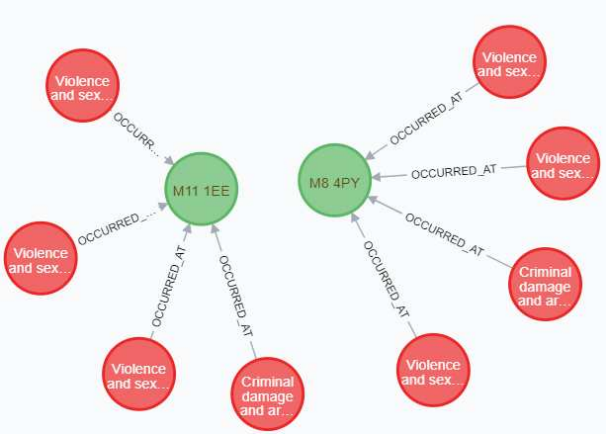


(b) Inexact (similarly threshold = 0.8) patient admission patterns with specific ICU type and a prescription

Figure 6.7: Individual measure for Mimic Data (MD)

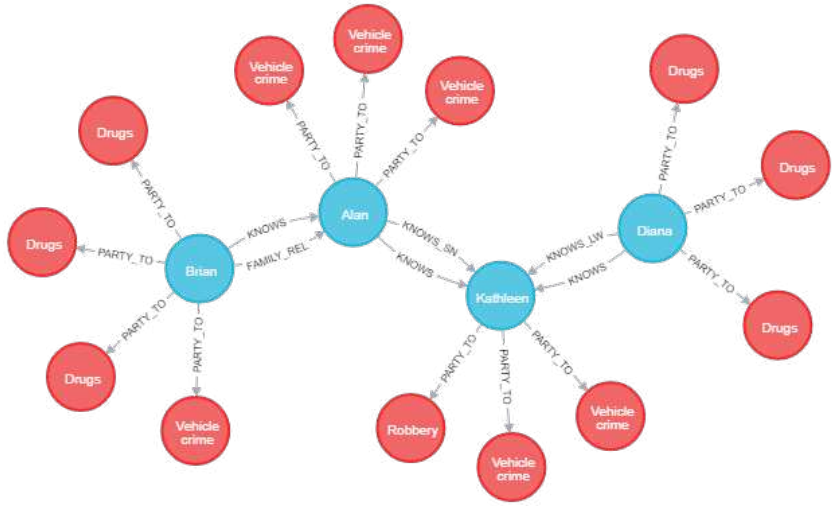
An exact match result of *neighborhoodSimilarity* identifies a group that collectively exhibits all the indicators in the query graph and marks it as suspicious for further investigation by law-enforcement. The customized Neo4j procedure also allows investigators to find suspicious groups that are not exact matches with the query graph by reducing the *similarityThreshold*. Figure 6.6c interprets an inexact match result (the social media details were truncated for easier visualization). All four individuals who know each other demonstrated suspicious activities: three have indicated 'Received training', 'Purchase weapons' and 'Suspicious travel'. It points to a group that must be investigated.

(a) Query graph for location 'OL10 2JL'



(b) Exact crime patterns by location based on the query graph

(c) Query graph for person 'Brian'



(d) Inexact (similarity threshold = 0.7) crime patterns by person

Figure 6.8: Individual and neighborhood measure for Crime Data (CD)

6.7.2 Patient's ICU stays data analysis

Figure 6.7a depicts a patients query graph. It consists of an 'EMERGENCY' admission, an ICU stay in 'MICU' type, and a prescription of 'Warfarin'. Therefore, with this query graph, we are looking for patients admitted to the 'EMERGENCY' section, then sent to the 'MICU', and prescribed 'Warfarin'. Figure 6.7b shows two of the inexact patterns (when similarity threshold = 0.8) that patients admitted the the 'EMERGENCY' section, then maybe stayed in 'MICU' and prescribed the drug 'Warfarin' in any of the admission. The patient having ID: 711 has five hospital admissions, and three times (out of 5 admissions), he/she was admitted to the 'MICU'. The drug

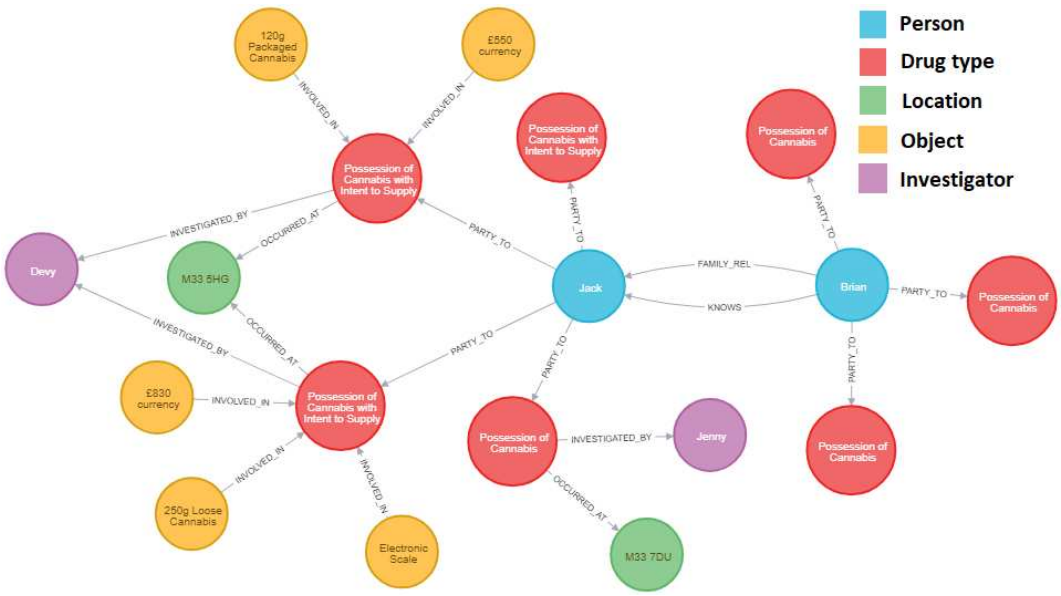


Figure 6.9: An identified drug network using neighborhood measure (CD)

'Warfarin' was prescribed twice when was in the 'MICU' (within one admission) and five times when was not in the 'MICU' (within one admission). Likewise, we can discuss the pattern of the patient (id = 249) who has three hospital admissions. These queries help to analyze the side effects of certain medications and patients' patterns of certain diseases.

6.7.3 Crime location and criminal analysis

Figure 6.8a shows the crime pattern for a location (postcode – 'OL10 2JL') that was retrieved as the query graph it consists of three 'Violence and sexual offenses' crimes and one 'Criminal damage and arson' crime. The bottom graph in Figure 6.8b shows some of the exact patterns for other locations based on the query graph. Figure 6.8c depicts the crime pattern of a criminal called 'Brian', which consists of 3 drug crimes and 1 vehicle crime. In response to a query based on a person's identifier, it shows his crime details and presents the crime pattern via the *implicitIndividualSimilarity*. Figure 6.8d shows the results when the similarity threshold is reduced to 0.7. The crime patterns of 'Alan', 'Kathleen', and 'Diana' are fetched as fairly similar crime patterns to that of 'Brian'. The database is also capable of fetching their relationship (if one exists) and identify whether they know each other. Such querying capability is highly important to investigators: iden-

Table 6.6: Query time of radicalization datasets **without** database caching. All queries are invoked for the first time (1st run) in a database instance. μ_q^{IS} and μ_q^{NS} denote the mean query time (ms) of *Individual Similarity (IS)* and *Neighborhood Similarity (NS)*. σ_q^{IS} and σ_q^{NS} denote the standard deviation of the query time (ms) of *Individual Similarity (IS)* and *Neighborhood Similarity (NS)*.

Dataset	Similarity Threshold	μ_q^{IS}	μ_q^{NS}	σ_q^{IS}	σ_q^{NS}
DS_1	0.6	244.4	343.7	30.5	20.0
	0.8	211.3	348.1	20.4	35.1
	1	211.4	348.8	26.5	34.9
DS_2	0.6	489.2	995.5	43.0	85.4
	0.8	462.3	1020.8	21.9	88.9
	1	473.2	905.1	66.4	52.8
DS_3	0.6	2352.9	5178.4	140.3	342.2
	0.8	2173.4	5166.0	107.5	319.2
	1	2191.0	5084.1	110.4	227.3
DS_4	0.6	28201.6	65958.0	2160.8	4501.1
	0.8	21170.0	55797.3	863.3	2333.2
	1	20886.6	49719.2	1342.5	4241.0

tifying others with similar crime patterns or those who have similar modus operandi, and identify connections among the criminals.

Example: Detecting drug networks

We next illustrate how the neighborhood measure helps in discovering group involvement in crimes. There are 2 charge types for drugs crimes, namely 'Possession of drugs' & 'Possession of drugs with intent to supply'. The query graph in this case contains different drug charge types and only 'drugs' as the crime type. Brian's three crimes (Figure 6.9) are drug related and charge type is 'Possession of Drugs'. He has family relationship with Jack; Jack was charged for both 'Possession of drugs' and 'Possession of drugs with intent to supply' and was found with 'packaged & loose cannabis'. Therefore, Jack may be flagged for investigation as a potential cannabis supplier for Brian. Furthermore, Jack was caught twice with cannabis in a certain location ('M33 5HG') and investigators can focus on offenders associated with that location to trace the drug network.

Table 6.7: Query time of radicalization datasets **with** database caching, 2,3,4, and 5 denote the consecutive runs of the same query. The details of the 1st run (without caching) are depicted in Table 6.6. μ_q^{IS} and μ_q^{NS} denote the mean query time (ms) of *Individual Similarity (IS)* and *Neighborhood Similarity (NS)*. σ_q^{IS} and σ_q^{NS} denote the standard deviation of the query time (ms) of *Individual Similarity (IS)* and *Neighborhood Similarity (NS)*.

		2				3			
Dataset	Similarity Thresh.	μ_q^{IS}	μ_q^{NS}	σ_q^{IS}	σ_q^{NS}	μ_q^{IS}	μ_q^{NS}	σ_q^{IS}	σ_q^{NS}
DS_1	0.6	55.9	124.0	12.8	15.0	41.1	101.0	4.4	22.3
	0.8	47.4	121.9	3.2	13.7	40.4	115.7	3.8	13.7
	1	50.0	123.8	5.0	15.4	44.5	107.8	7.5	15.1
DS_2	0.6	226.0	602.5	20.2	27.0	199.2	536.0	13.6	34.5
	0.8	227.2	592.5	12.6	36.1	180.9	544.5	21.9	39.5
	1	230.8	571.7	19.4	32.1	170.0	526.7	13.5	37.4
DS_3	0.6	1809.2	4668.9	132.3	306.4	1725.0	4397.6	196.0	284.9
	0.8	1677.0	4585.2	121.3	353.4	1628.2	4494.7	81.0	362.0
	1	1687.6	4409.6	120.2	335.1	1620.3	4275.4	107.8	376.1
DS_4	0.6	28478.4	68780.7	1493.5	1775.4	28689.8	67533.0	1522.5	3252.5
	0.8	21563.6	57595.3	542.2	1641.5	21475.8	53957.7	807.2	2409.4
	1	20485.6	50618.0	1382.0	4795..0	20347.0	48364.4	1297.6	6312.0
		4				5			
Dataset	Similarity Thresh.	μ_q^{IS}	μ_q^{NS}	σ_q^{IS}	σ_q^{NS}	μ_q^{IS}	μ_q^{NS}	σ_q^{IS}	σ_q^{NS}
DS_1	0.6	29.6	113.5	4.3	23.0	26.7	78.9	2.9	10.7
	0.8	25.8	111.8	4.1	19.4	22.2	80.2	0.8	13.3
	1	25.2	98.3	3.6	15.8	22	80.9	2.0	16.9
DS_2	0.6	207.5	528.8	20.3	27.1	190.9	543.1	23.2	33.8
	0.8	194.5	530.0	22.5	39.5	175.9	516.9	22.6	40.9
	1	177.4	527.1	16.4	39.5	157.9	518.9	12.4	36.8
DS_3	0.6	1666.7	4403.2	143.1	284.9	1650.7	4375.6	127.1	306.3
	0.8	1574.9	4471.8	75.4	304.6	1532.7	4426.0	81.1	379.3
	1	1617.6	4297.6	104.5	368.25	1579.2	4203.9	97.3	343.5
DS_4	0.6	28705.6	67582.3	1245.8	3170.9	28854.0	67020.7	1589.2	3543.3
	0.8	21395.4	54780.3	780.9	4687.8	21219.0	54492.0	902.3	1676.5
	1	20385.8	48582.2	1659.2	6475.7	20848.0	48744.0	907.8	5794.8

6.7.4 Query performance tests

The query performance experiments of the PINGS library determine the efficiency and performance capabilities via graph databases w.r.t. the previous investigative searches (B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019). We ran query performance tests on Neo4j graph databases for radicalization data and also inspected the cache performance of databases by different sizes of the datasets. The query graph shown in Figure 6.3 is utilized in all tests, which enables the comparison among datasets. We used a machine with Intel Xeon(R) 3.30GHz CPU and 64GB RAM. We generated radicalization datasets of varying sizes using our data simulator. The graph density is maintained in each case where averaged 3 indicators per persons. Table 6.5 shows the radicalization datasets' details. μ_q^{IS} and μ_q^{NS} denote the mean query time of *Individual Similarity (IS)* and *Neighborhood Similarity (NS)* respectively. σ_q^{IS} and σ_q^{NS} denote the standard deviation of the query time of *Individual Similarity (IS)* and *Neighborhood Similarity (NS)* respectively. We consider three exact match (*similarityThreshold* - 1) and two inexact match (*similarityThreshold* – 0.6 and 0.8) scenarios. Table 6.6 shows the query time without utilizing the database cache as a procedure is executed for the first time in a database instance. Figure 6.10 depicts the mean query time of radicalization datasets in the 1st run without making use of the database cache. *neigborhoodSimilarity* searches for all possible group combinations and hence takes more time than *individualSimilarity* which just evaluates individuals. When the dataset size increases, the query time difference between the exact and inexact matches also increases. This is expected as the number of group combinations to inspect in the *neigborhoodSimilarity* increases significantly with the increase in the number of persons and their relationships.

Table 6.7 shows the mean query time with the database cache in consecutive runs (up to 5) of the *individualSimilarity* and *neigborhoodSimilarity*. Figure 6.11 illustrates the mean query time with the database cache over query runs for each dataset. Database caching is one of the key features in any type of database to improve the query performance. In-memory data caching can be one of the most effective strategies for improving the overall application performance and reducing your database costs (Amazon Web Services (AWS), 2020). As depicted in Figure 6.10, the 1st run

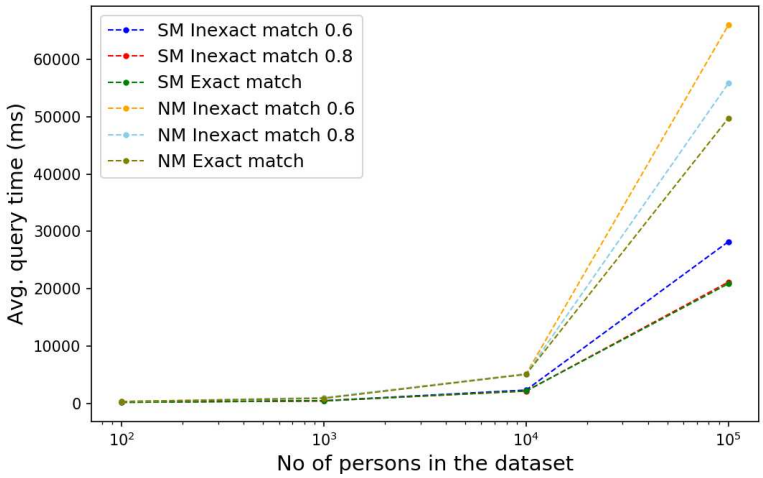


Figure 6.10: Mean query time (ms) vs. no of persons involved in a dataset (Table 6.5) for the 1st run (without database cache).

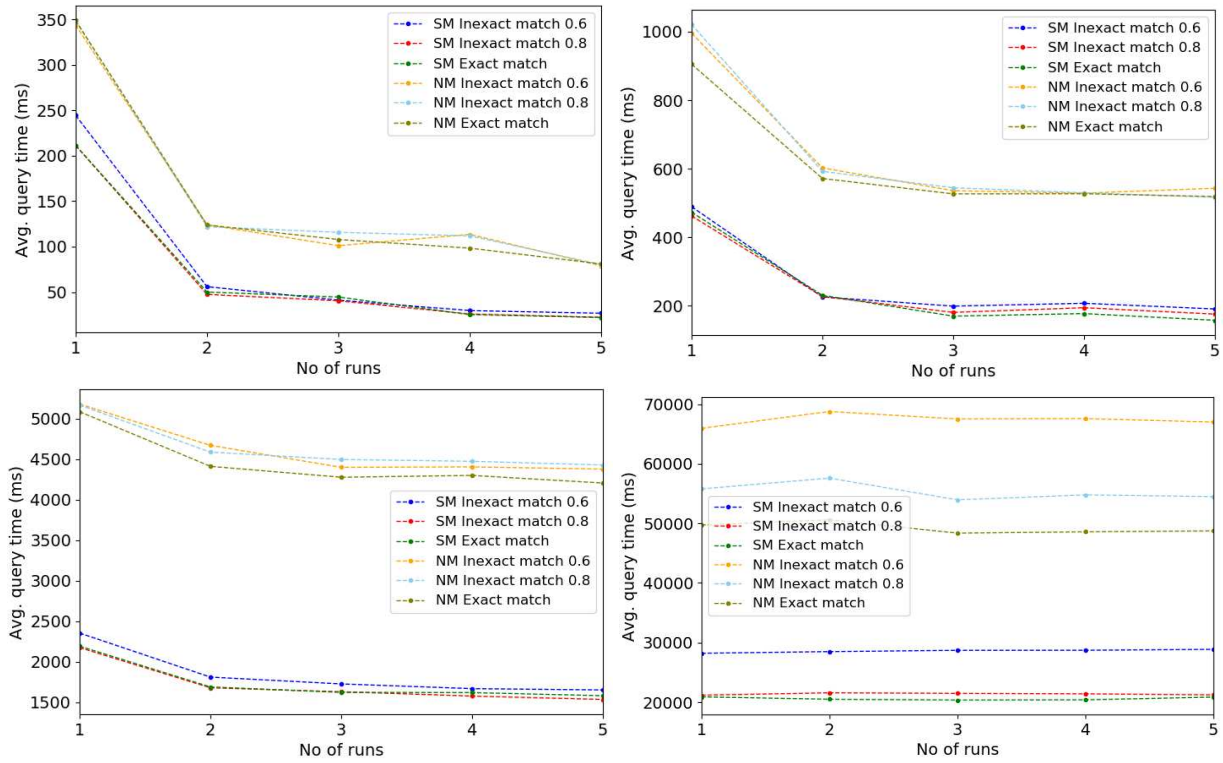


Figure 6.11: Mean query time (ms) vs. number of consecutive query runs (with the database cache) for each dataset. The details of the datasets are shown in Table 6.5. DS_1 [upper left], DS_2 [upper right] DS_3 [bottom left], and, DS_4 [bottom right].

represents the query time without caching, when the query is requested for the first time after a database instance has started. For the next runs, we can see a significant query performance as many of the frequently fetched data stores in the in-memory cache. For example, inexact matching

(*similarityThreshold=0.8*) for DS_1 took $211.3s$ without database caching (Table 6.6), while the initial run with database caching only took $47.4s$ to complete the same query (Table 6.7). Over the dataset sizes, the cache performance reduces due to the limitation of the size of the in-memory cache. However, many databases allow users to adjust the cache size, and Neo4j too provides flexibility to change the size of the in-memory cache which enables the maximum query optimization (Neo4j.com, 2020b). Figure 6.11 depicts that the smallest dataset (DS_1) preformed well with the database cache and also the standard deviation is generally reduced over database runs (Table 6.7), indicates the stability that provided by database cache. Thus, graph databases enhance the query capabilities and the performance of investigative searches by adding database features.

Exploration and mining of behavioral data patterns in large datasets in the form of networks or graphs that are continually evolving is an important task in many problem domains. The data sets are incomplete and contains massive amounts of non-related information, making querying further complicated. We propose a solution based on graph databases and inexact pattern matching for investigative pattern detection. Our enhanced solution also includes a novel similarity measure for optimized node and edge matching with bidirectional query graphs. The proposed technique facilitates retrieval of individuals and groups of interest based on the similarity to a complex query pattern. The mechanisms developed are made available in the form of an open-source library of routines, PINGS (Procedures for Investigative Graph Search), implemented as several custom procedures for Neo4j (Computer Network Research Laboratory (CNRL), Colorado State University, 2022g).

Synthetically generated radicalization datasets of different sizes and a real crime dataset were imported as graph databases and used to validate and evaluate the performance of the investigative pattern detection procedures. Results demonstrate the capabilities of the proposed technique and the inexact similarity scoring mechanism to discover potential individuals, groups, and patterns of interest. The impact of Neo4j database caching technique to enhance the query performance, which is crucial in investigations that typically involve iterative querying by a human in the loop

to narrow down the solution space, was depicted by invoking custom queries multiple times in a database instance.

The following section discusses the further enhancements to the PINGS library that utilize the timestamps to weight sequences of events based on their recency and repeated occurrences of the same indicators. Such weighting and compensation mechanisms allow investigators to customize their queries more accurately and flexibly based on the needs.

6.8 Approaches to enhance investigative graph search

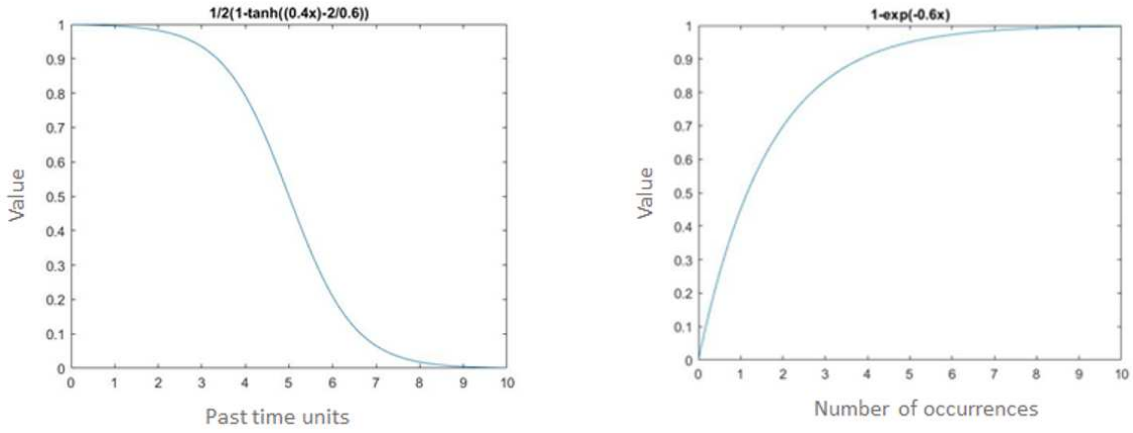
In investigative domains, evaluating the relevant temporal data is time-critical to detect recent advancements of specific trends required to take immediate actions. In the radicalization context, some individuals and groups may be active in irregular time intervals. Therefore, the time-based evaluation leads to prevent any threats proactively and efficiently. In the data extraction, most of the behavioral indicators attached to a timestamp that describes the time of occurrence of particular event. In this version, We utilize that temporal information to enhance graph search algorithms.

Furthermore, an individual may exhibit a behavioral indicator multiple times. In such cases, the algorithm has to tailor that significance over the multiple occurrences. However, after some point, the significance has to be saturated because we already captured the contribution of that particular behavioral indicator, and also we look for the collective measure over different indicators based on a given query graph. Therefore, having multiple occurrences of indicators need to be precisely handled in investigative domains.

$$h(t, a, b, c) = \frac{1}{2}(1 - tanh(\frac{at - b}{c})) \tag{6.4}$$

$$g(x, d) = 1 - e^{-dx} \tag{6.5}$$

This enhanced version of the PINGS library introduced two different functions to manipulate timestamps and multi-occurrences. The primary idea was captured from the initial work



(a) An example Sigmoid function to assess the recency of a behavioral indicator based on time of occurrence

(b) An example exponential function to compute the multi-occurrences of a behavioral indicator

Figure 6.12: Functions to evaluate the recency and multi-occurrences

in (B. W. Hung, Jayasumana, & Bandara, 2019). Figure 6.12a depicts an example Sigmoid function in Eq. 6.4, we used to weigh the timestamps. The Sigmoid function fits well in this case because it initially maintains the highest weight to a notable time-period and exponentially reduces the weight over time. By altering three input parameters of the Sigmoid function, we can use various shapes that fit the different datasets and requirements. Figure 6.12b shows an example exponential function in Eq. 6.5 that we used to capture the number of occurrences. This type of exponential function produces the required functionality in investigative domains. When the number of occurrences increases, the significance (weight) of the particular indicator also exponentially rises. However, after a certain number of occurrences, the weight saturated, which means the measure already captures that indicator's significance maximumly. According to the need, we can change the input parameters of the function to obtain various shapes that differ the saturation point.

With the addition of these two functions, the input parameters of procedures have become complicated. We realized that many of the parameter values are data-specific and no need to change frequently. Therefore, we added all the parameters except the similarity threshold to the configuration list. Figure 6.13 depicts a configuration file with all the elements. In *funcionalParameters* is a 4-elements of array that consists of $[a, b, c, d]$ input parameters in Eq. 6.4 and 6.5.

```
{
  "nodes": [
    {
      "label": "Person",
      "attrs": []
    },
    {
      "label": "Indicator",
      "attrs": ["name"]
    }
  ],
  "relationships": [
    {
      "fromNode": "Person",
      "toNode": "Indicator",
      "queryRelType":"EXIBHITS",
      "dataRelTypes": ["EXIBHITS"]
    }
  ],
  "queryLabelName":"Query",
  "queryFocusLabelName":"Person",
  "indicatorLabelNames": ["Indicator"],
  "functionParameters":[0.4, 2, 0.6, 0.8],
  "timestampRelationshipTypes": [
    {
      "relType": "EXIBHITS",
      "relAtrrName": "date"
    }
  ],
  "timestampFormat": "mm/dd/yyyy",
  "latestTimestamp": "01/01/2020",
  "recentnessYearRange":10
}
```


Figure 6.13: An example configuration list in PINGS v2.0

In this version, we additionally appended the metadata for relationship types. *fromNode* and *toNode* defines the expected label types with the direction. In some datasets, multiple relationship types between two specific nodes need to be matched. However, a query graph defines a single relationship type. In such cases, *queryRelType* denotes the relationship types in the query graph and *dataRelTypes* can have all the relationship types in the data graph required to match. Like already mentioned, the timestamp was attached as a property to a relevant relationship type. *timestampRelationshipTypes* defines all the relationship types along with the relevant property name (*relAtrrName*). Timestamp format can be defined as well. *latestTimestamp* is the reference timestamp to calculate the score of a given query graph. *recentnessYearRange* defines the length of the x-axis in Figure 6.12a. Therefore, the configuration files consist of all the dataset-specific details to execute the enhanced PINGS library. Figure 6.14 depicts a configuration list utilized for the Western Jihadism Project (WJP) graph database.

Figure 6.14: An example configuration list for WJP graph database

With all these modifications to the configuration list, the input parameters in Neo4j procedures were simplified. In this version of the PINGS library, *similarityThreshold* is the only input parameter in the procedures, which is expected to change continually. Therefore, the procedures in PINGS v2.0 were designed as follows.

$$individualMeasure(similarityThreshold) \tag{6.6}$$

$$neighborhoodMeasure(similarityThreshold) \tag{6.7}$$

6.9 Rel2Neo: Relational to Neo4j graph database conversion

With the rapid improvement of data collection technologies and social networks, highly linked, rich-informative data is widely available. However, the graph databases have been evolved very recently, and much of the information is still stored in relational databases. As explained in Section 6.4 graph databases were designed to treat the relationships between data as equally important as the data itself. Therefore, storing highly-linked data in graph databases can be significantly beneficial for efficient data querying and analyzing various insightful data patterns. Nevertheless, there is no appropriate tool or software to convert a relational database to a graph database to the best of our knowledge.

Therefore, we implemented a Python library to efficiently convert a relational database to a graph database (called 'Rel2Neo'), which was made available as an open-source library under GPL-3.0 License (Computer Network Research Laboratory (CNRL), Colorado State University, 2022i). Initially, we developed the library to convert the Western Jihadism Project (WJP) SQL-based relational database into a graph database in Neo4j in order to explore the power of graph data science in driving new radicalization insights. In the implementation, we verify the library was written in a standard structure where applicable to any relational database for conversion in a controllable way. The current version of the library supports the CSV file format, which is widely available in various relational databases and graph databases. Figure 6.15 depicts for the process for transforming the relational database to the graph database.

Initially, we import all the tables as separate CSV files. Then, we have to write a configuration file in JSON format to describe the conversion programmatically. Figure 6.16 depicts an example file snippet of a configuration file. The file allows the full capability to generate the required graph structure and customizable based on the user requirement. It allows defining the nodes and edges as needed and controlling the attributes and the datatypes of attributes. The foreign keys are

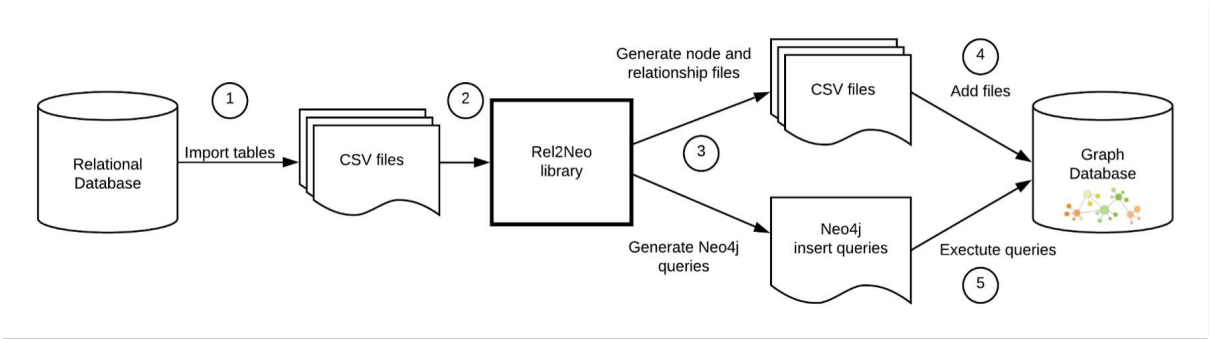


Figure 6.15: Relational to graph database conversion (Rel2Neo) pipeline

implicitly considered to implement relationships to the graph database. The configuration file also allows defining the direction of the relationship type.

```
{
  "file_name": "person.csv",
  "file_alias": "",
  "name": "Person",
  "isNode": true,
  "indexes": ["person_id"],
  "data_types": [
    {"int": ["person_id","legacy_id","legacy_id_num","year_first_terror_le_contact",
             "year_born","year_radicalization","year_death"]},
    {"float": []}
  ],
  "skips":["ethnicity_id","radicalization_reason_id"],
  "foreign_keys": [
    {
      "id":["country_id","country_id"],
      "from_id":["person_id", "person_id"],
      "from_table":"Person",
      "to_table":"Country",
      "data_type":"int",
      "name":["COUNTRY_OF_RESIDENCE", false],
      "attr": []
    },
```

Figure 6.16: An example code snippet of the configuration file for database conversion in Rel2Neo

After the implementation of the configuration file, the 'Rel2Neo' library is capable of generating CSV files for both nodes and relationships and the insert queries to the Neo4j graph database. If we define indexes (attributes have unique values) in the configuration file, the library will also generate the index insert queries in the same query file. Then the generated CSV files via the library are placed in the 'import' folder of the graph database. We have implemented a Python script

to execute all the generated quires sequentially. With just one click of execution, the WJP graph database is established. The graph schema of the WJP database is shown in Figure 12. Further, the database conversion library can convert any relational database to a graph database in a fully controllable way.

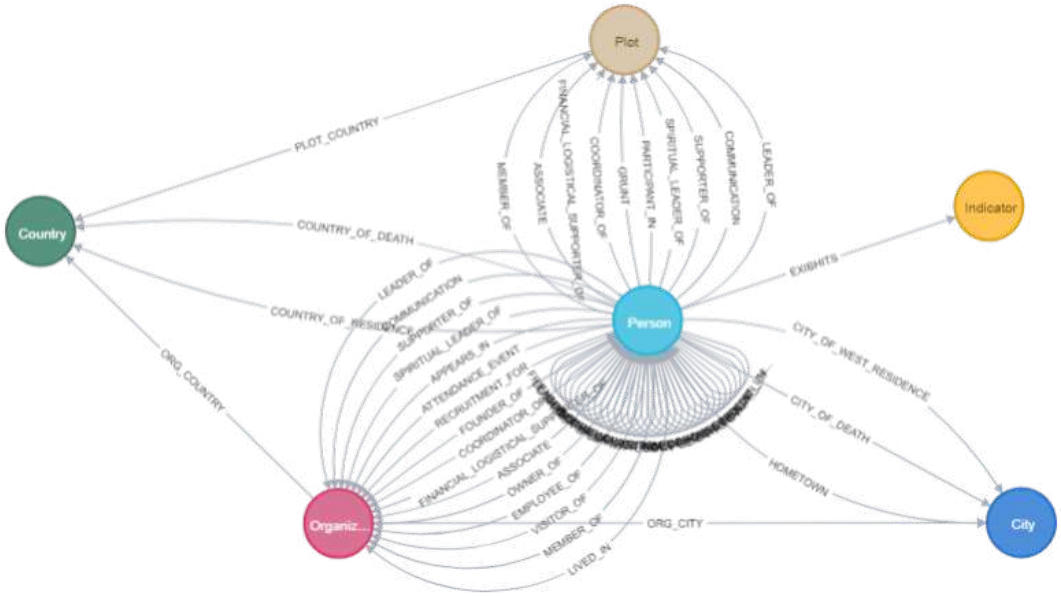


Figure 6.17: Graph schema for the Western Jihadism Project (WJP) graph database

We continued to assess the myriad of analysis and visualization opportunities in the Neo4j environment that would be helpful for social scientist researchers and law enforcement analysts. We implemented a simple web application with the Leaflet library (Agafonkin, 2020) as a way to geospatially display persons from the WJP database onto a map using their city of west residence, hometown, and city of death. Figure 6.18 and 6.19 illustrate the screenshots of the web application for WJP individual's city of west residences and hometowns, respectively. The size of the circles (colored in red) is proportional to the density of the particular location.

Figure 6.18: Map of WJP individuals showing their city of west residence

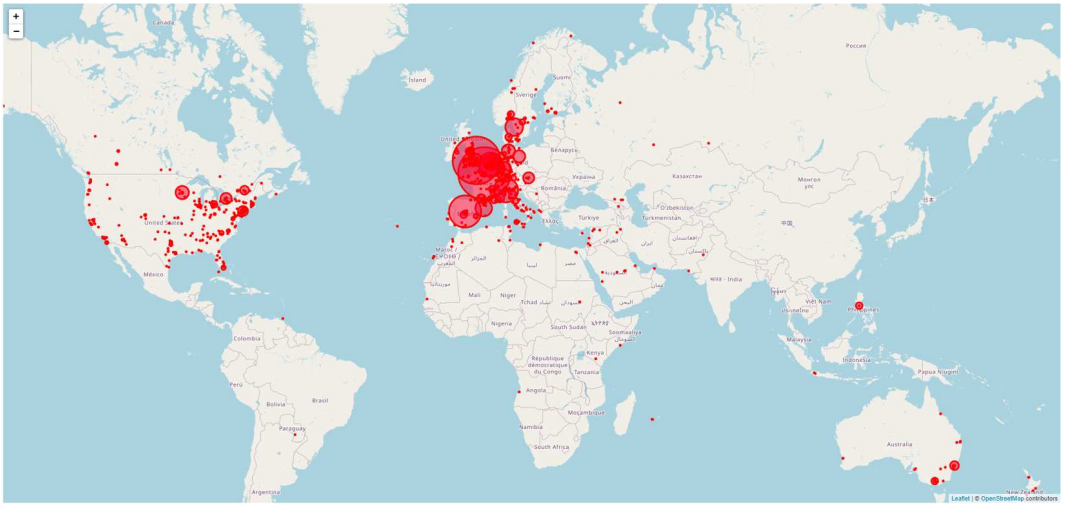


Figure 6.19: Map of WJP individuals showing their hometown

Chapter 7

A comparative study of complex data object generation with likelihood and deep generative approaches

7.1 Introduction

Machine learning techniques have emerged that rely on large sets of well-defined and formatted data for automated learning. Such techniques are widely used for applications ranging from commerce (Lee & Shin, 2020) to computer security (Handa, Sharma, & Shukla, 2019). Essential for these techniques are collections of data and data sets, both public and proprietary. Thus, mining of data such as those associated with social networks, online purchases, and personal interactions have become increasingly important. However, compilation of formatted and consistent data sets is often not possible in fields such as social sciences, criminology, and public health. In contrast, with conventional ML applications such as image classification and face recognition, the data sets typically have a well-defined form with all the values (e.g., set of pixels in the image) available at least with training data. However, our focus here is those applications where some data points are unavailable even within smaller data sets, either because they were not measured, not known, do not exist, or simply are not applicable. In this work, such datasets are defined as Unconventional Data Sets (UDS).

UDS are common in domains such as social, political, and health sciences. Certain data sets (e.g., (Klausen et al., 2020)) are collected from public information sources such as court documents and newspapers, which contain only the information discovered by reporters or law enforcement, but they are not guaranteed to be complete. The absence of value of a behavioral indicator for a person, e.g., date of marriage, does not rule in or out the existence of the indicator but only that it is not known (B. W. Hung et al., 2018; Muramudalige et al., 2019). Demographic data

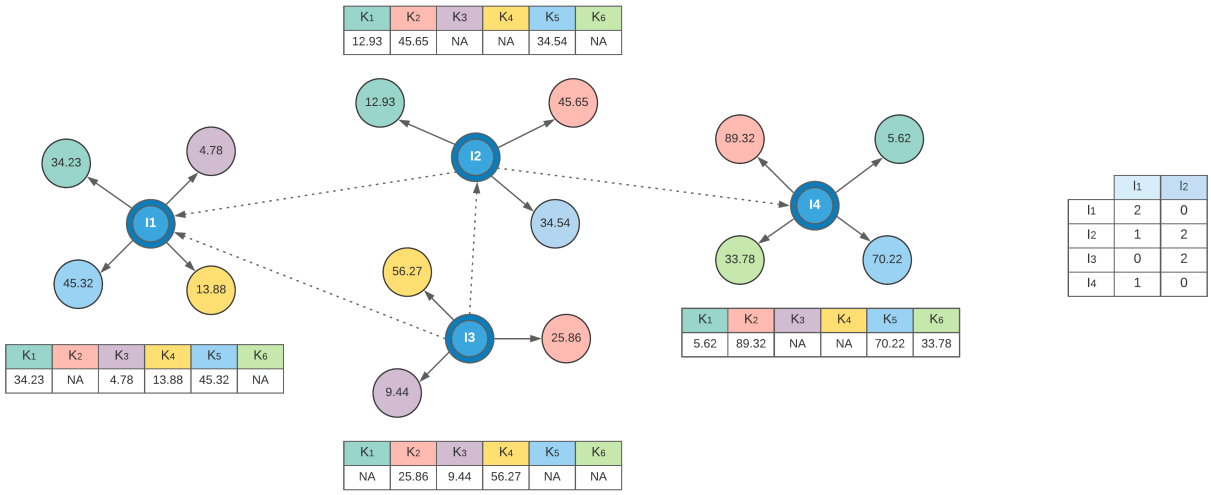


Figure 7.1: An example social, behavioral data network. Initially, each entity (blue nodes) are mapped to star data objects. Then, the hierarchical structure are mapped based on the adjacency matrix in the right-side

sets often contain information gathered from different surveys that may contain overlapping and non-overlapping queries resulting in inconsistencies and missing data. While datacentric research relying on the availability of heterogeneous and complex formatted data forms the backbone of many research domains, there are many impediments to applying such techniques in other domains where UDS are the norm.

Further, Unconventional Data Sets (UDS) such as social, behavioral networks can also be interpreted as a collection of complex data objects belonging to classes of data objects such as images, trees, and graphs. A major challenge of this representation is that the data often naturally lie in non-Euclidean spaces (H. Wang, Marron, et al., 2007). This has given rise to several approaches for object oriented data analysis to perform statistical analysis of populations of complex objects (H. Wang et al., 2007; Aydın et al., 2012; Sienkiewicz, Wang, et al., 2018). In social and behavioral data, e.g., in Figure 7.1, the simplest form of the objects are star networks, which depicts a set of activities/indicators (leaves) connects for an entity (root).

A related problem in these domains of interest to us is the need to generate synthetic data that mimics the existing data, e.g., to meet privacy/confidentiality constraints, to increase the volume of data or to obtain data sets with consistent records. A few instances where synthetic data sets play a significant role are outlined next. Use of personal information is subject to severe restrictions

related to the privacy and confidentiality of data that often prevents sharing and even publication of results. Here, synthetic data can capture the underlying structure of data while preserving the privacy of the sources. A second scenario is when the amount of data available is inherently limited and/or some data fields unavailable due to collection challenges. In radicalization profile analysis, e.g., social scientists rely on text sources such as official press releases, court documents, trusted news sources, and verified social media accounts of extremists. The researchers read and inspect related text documents and then manually label the behavioral indicators present (a process known as 'coding'). Not only is the number of potential documented cases very limited, but the cost and effort to create appropriate records is quite high, resulting in small datasets. Furthermore, some data fields are unavailable due to the fact that different records are gathered from different forms of documents and different sources, resulting in UDS. A consequence of such small and UDS is that it limits the ability to use state-of-the-art techniques such as ML (Hung et al., 2019; Gianfrancesco, Tamang, Yazdany, & Schmajuk, 2018) for detection and profiling. Further, it drastically limits the amount of material available to train individuals to carry out coding. There is thus a critical need for techniques to capture and learn the data distribution from small and/or unconventional sets of data and synthesize larger and/or specialized data sets as needed.

In this work, we make two major contributions. First is a feature mapping technique to statistically model UDS as complex data objects. Here, we focus on UDS that are small and sparse with a considerable number of unavailable entries. The proposed feature mapping technique shows the ability to overcome data-specific challenges with UDS such as unavailable entries thus facilitating the use of an array of ML algorithms. We illustrate its broad applicability using UDS in radicalization, programmer performance, and medical records. Second contribution of this research is the generation of synthetic data from UDS. The proposed feature mapping technique is applied with two likelihood methods and a deep generative method for different data objects and compare their performance. As likelihood methods, we use Gaussian and regular-vine copulas. We use an Adversarial Autoencoder (AAE) (Makhzani et al., 2015) as the deep generative approach because of its fluency in generating high-fidelity and discrete distributions (Shirazi et al., 2020). A simulation

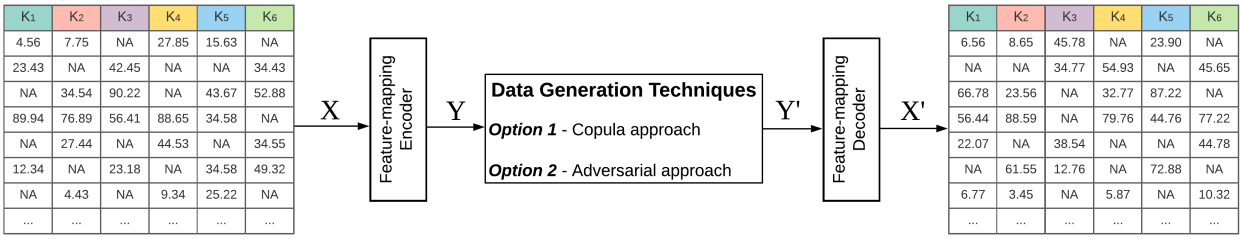


Figure 7.2: Data generation pipeline for sparse and inconsistent key-value datasets.

and three real-world datasets verify the efficiency and robustness of each generative approach over different data unavailable rates in UDS. We also apply the same techniques to generate hierarchical structure of a given network where allows handling complex network topologies. The impact of this work facilitates larger datasets for use cases in diversified domain by synthesizing UDS using the proposed feature mapping technique. Our work further ensures some degree of privacy because the synthesized UDS are different and novel while following the underlying actual distributions.

7.2 Data representation

In the following, we present the details of the three real UDS used for experiments and discuss the applicability with data representation in Figure 7.1.

7.2.1 Datasets

Radicalization dataset. The Western Jihadism Database (Klausen et al., 2020) has almost all the incidents of terrorist actions committed in western countries including timestamps and yet it does not consist of unconfirmed or undiscovered activities that yield a sparse dataset. Data collected from public sources including court documents, analyst reports, and newspapers, may be incomplete, which leads to an unconventional dataset. In this work, we use 135 detailed pathways of home-grown jihadists (Klausen, Libretti, et al., 2018) extracted from radicalization trajectories of 335 known American jihadists (Klausen et al., 2016) over 24 behavioral indicators. Data representation as shown in Figure 7.1, blue nodes denote extremists, and leaf nodes depict respective behavioral indicators connected to extremists. Different colors in leaf nodes represent various

behavioral indicator types, while node values indicate a value corresponding to the particular indicator's time of occurrence. The connections between blue nodes illustrate the relationships among extremists.

Stack overflow dataset. Stack Overflow[8] is a question answering website which attains badges to encourage user engagement and guide behaviors. We used publicly available archived dataset[9] related to data science. Since this is a class of behavioral pattern data, not all the badges are gained by each user. Therefore, the dataset is intrinsically unconventional, and we extracted 285 records across 32 badges from the actual dataset. By mapping to the data representation in Figure 7.1, each blue node representing a user and different badge types denotes the colored leaf nodes. The numerical values in leaf nodes represent a value related to the badge accomplished time. Links between blue nodes indicate relationships among users characterized based on similar user attributes (e.g., age range, profession, etc.).

Mimic III dataset. MIMIC III medical dataset (Johnson et al., 2016) is a large, freely available database of clinical visit records of Intensive Care Unit (ICU) patients between 2001 and 2012. In this research, we use SERVICES data table[10] that describes services that patients were admitted under. We extracted 500 patient records across 6 different service types. Even though there are 6 service types, patients may not take all of them, leading to an unconventional dataset. With the given data representation in Figure 7.1, blue nodes represents the patients and leaf nodes denote service types. Connection between blue nodes illustrate the different relationships among patients.

Figure 7.1 depicts a general snapshot of social or behavioral data (a type of UDS). Blue nodes represent entities, e.g., persons or any other objects. Other nodes (leaf nodes) connected to entities denote a related type or a category or a key. Different types are depicted with different colors as shown in Figure 7.1. Leaf nodes also have a numerical value related to a given type (key) in the data. Therefore, a leaf node consists with a key and a numerical value that enables to map the

[8]https://stackoverflow.com/

[9]https://archive.org/download/stackexchange

[10]https://mimic.physionet.org/mimictables/services/

data in a leaf node as a key-value pair. However, there may be hierarchical topological structures in the networks that connect nodes together (dashed lines in Figure 7.1) as explained in previous data sets. In this work, we particularly interest the data in leaf nodes that consist of both key and a numerical value. Therefore, we consider each entity independently where the simplest form of the object is a star network and descriptively explains in Subsection 7.3. Later, we consider hierarchical typologies and apply same data generation techniques to generate hierarchical data. We discuss the relevant experiments and results in Subsection 7.5.1.

As aforementioned, these types of data are inherently unconventional, because each entity does not contain all the available keys as shown in tables in Figure 7.1 for each star network. The unavailable keys are denoted as 'NA'. Therefore, mapping data to key-value pairs is not straightforward and need the maintain the information of availability of each key. The data modeling comprehensively discussed in Subsection 7.3. We simulate the proposed approach in Section 7.4 and further validate through 3 real-world UDS in Subsection 7.5.

7.3 Statistical modeling

7.3.1 Complex data objects

The key-value pair (KVP) data structure is a commonplace in well-formed and complete datasets as explained in Section 7.1. However, in many practical scenarios, the data is unconventional due to diverse reasons that provide sparsity and inconsistency towards data modeling. In complex object modeling, KVP data structure still beneficial to define the datasets for precise analysis.

Generally, a key-value pair can be denoted as (k, v_k), where v_k is the value of the key k. Often, when v_k is not available, a "NA" will be assigned. Here, we introduce δ_k to indicate the availability of the value v_k. That is, $\delta_k = 1$ if v_k is available, and 0 if v_k is not available. Then, a data point can be denoted as (k, v_k, δ_k), and the ith data object can be written as

$$o_i = \{(k, v_k, \delta_k), \forall k; k \in \mathcal{K}\}, \tag{7.1}$$

where $\mathcal{K}$ is the collection of all possible keys. An example is given in the data table on the left in Figure 7.2, the 1st data object (o_1) is shown as $\{(k1, 4.56, 1), (k2, 7.75, 1), (k3, NA, 0), (k4, 27.85, 1), (k5, 15.63, 1), (k6, NA, 0)\}$.

In this paper, we are interested in the feature space of the datasets to apply the proposed feature-mapping techniques for complex object generation. Therefore, we also define the dataset w.r.t to the features (columns in the dataset). Let X_k to be the kth column. For instance, the first column (X_1) can be expressed as $\{(k1, 4.56, 1), (k1, 23.43, 1), (k1, NA, 0), (k1, 89.94, 1), (k1, NA, 0), (k1, 12.34, 1), (k1, NA, 0), \dots\}$. Thus, the data table can be indicated as

$$X = \{X_k : k \in [1, K]\} \tag{7.2}$$

where K denotes the total number of columns (the size of $\mathcal{K}$). In this paper, we assume that $\mathcal{K}$ is known.

Let $\mathcal{X}$ denote the space of all possible features for a given $\mathcal{K}$. We further assume v_k takes positive values for $k \in \mathcal{K}$. The conditional distribution function of v_k given $\delta_k = 1$ is denoted by $F_{k,1}(\cdot)$.

7.3.2 Feature-mapping technique

To model the complex object data, values that are not available need to be properly addressed. Furthermore, for many applications in social and behavioral domains, the values (v_k) are widely spread among different value ranges that mitigates the identification of underlying distribution precisely.

The proposed feature-mapping techniques transforms data from $\mathcal{X}$ to a well-structured space $\mathcal{Y}$ that mitigates inconsistencies of the data and enables applying complex analytics and machine learning techniques. Therefore, we transform a given value v_k to w_k where

$$w_k = \begin{cases} F_k(v_k) & \delta_k = 1 \\ 0 & \delta_k = 0, \end{cases} \tag{7.3}$$

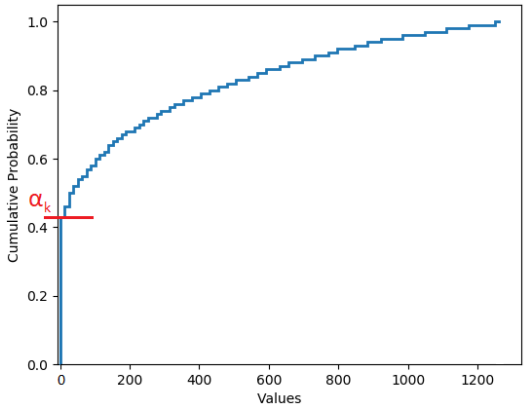


Figure 7.3: An example CDF of $F_K(.)$

and F_k is a non-decreasing function.

One choice of F_k is

$$F_k(z) = \alpha_k + (1 - \alpha_k)F_{k,1}(z), z \geq 0$$

where α_k denotes the probability of $\delta_k = 0$. Note that $F_k(z)$ is the cdf of v_k when replacing "NA" by 0. Figure 7.3 shows an example of $F_k(\cdot)$. In general, F_k is not continuous at 0.

In practice, we introduce a *data approximation technique* when $\delta_k = 0$; particularly,

$$w_k = \begin{cases} F_k(v_k) & \delta_k = 1 \\ u_k & \delta_k = 0, \end{cases} \tag{7.4}$$

where $u_k \sim U[0, \alpha_k]$. That is, instead of replacing "NA" with 0, here we assign a random number from a uniform distribution $U[0, \alpha_k]$. Let space $\mathcal{Y}$ denote the collection of elements in $\mathcal{X}$ whose key values satisfying (7.4).

We implement a *feature mapping encoder* to apply the feature mapping techniques that transforms data to a structured space ($\mathcal{X} \to \mathcal{Y}$). The feature mapped dataset can be denoted as Y similar to (7.2). Figure 7.2 shows our data generation pipeline for structured data. We use either likelihood (parametric) or deep-generative approach to generate data after applying feature mapping techniques. The details of the likelihood approach and deep-generative approach are discussed in subsections 7.3.3 and 7.3.4 respectively. Let the generated structured data denotes $Y' \subseteq \mathcal{Y}$ with

the assumption of the successful data generation where $Y' \simeq Y$. After the data generation, we use a *feature mapping decoder* that converts data back to the actual data space ($\mathcal{Y} \rightarrow \mathcal{X}$) and that particular dataset is denoted as X'.

In Subsection 7.4, we examine the performance of our proposed data generating framework. We use various decriptive statistics to validate the similarity between actual and generated data sets.

7.3.3 Copula approach: multivariate Gaussian and R-vine copulas

A copula is a probabilistic approach that captures the entire dependence structure of multivariate distribution for given marginal distributions. Copulas have been widely used for probabilistic data modeling and generation (Cherubini et al., 2016; Genest et al., 2009; Jammazi & Reboredo, 2016; X. Zhang & Jiang, 2019; Z. Wang et al., 2017; Zhao et al., 2019; Panamtash et al., 2020). There are many popular choices, e.g., Univariate, Archimedean (bivariate), and multivariate copulas to model data distributions.

In this work, we use two multivariate copulas for our data generation, i.e., Gaussian and regular vine (R-vine) copulas. The vine copula is known to be efficient in capturing complex dependencies that cannot be coupled by a Gaussian dependence structure (Torre et al., 2019). The vine copula has different types, and we select regular vine (R-vine) copulas, which have more choices of decomposing dependence structures than drawable vine (D-vine) copulas and canonical vine (C-vine) copulas (Wu et al., 2015).

Let a continuous random vector $(Y_1, Y_2, \ldots, Y_K)$ that includes all the columns of feature-mapped object data (Y) as shown in Figure 7.2. Then, a copula (C) of $(Y_1, Y_2, \ldots, Y_K)$ is defined as,

$$\begin{aligned} C(y_1, y_2, \ldots, y_K) = P(Y_1 \leq F_1^{-1}(y_1), Y_2 \leq F_2^{-1}(y_2), \\ \ldots, Y_N \leq F_K^{-1}(y_K)) \end{aligned} \tag{7.5}$$

where $F_i^{-1}(y_i)$ is the inverse CDF of ith column vector.

Multivariate Gaussian copula

Gaussian copula is constructed from a multivariate normal distribution over the space $\mathcal{Y}$. A Gaussian copula for a given correlation matrix $\mathcal{R} \in [-1, 1]^{K \times K}$ can be defined as

$$\mathcal{C}_{\mathcal{R}}^{Gauss}(y) = \Phi_{\mathcal{R}}(\Phi^{-1}(y_1), \ldots, \Phi^{-1}(y_K)) \tag{7.6}$$

where Φ^{-1} is the inverse cumulative distribution function of a standard normal, and $\Phi_{\mathcal{R}}$ is the joint cumulative distribution function of a multivariate normal distribution with zero mean vector and covariance matrix is same as the correlation matrix $\mathcal{R}$.

R-vine (regular vine) copula

Vine copulas generally apply when data distributions are complex that unable to be modeled with standard parametric distributions such as the multivariate Gaussian distributions, because of the nonsymmetric or heavy tail dependencies between some variables (Czado, 2011; Z. Wang et al., 2017). Vine copulas are a flexible class of dependence models with bivariate building blocks (Joe, Li, & Nikoloulopoulos, 2010). A vine is a graphical representation of labeling constraints as a set of tress in high-dimensional probability distributions. R-vine copula contains a sequence of nested trees T with nodes N and edges E. In R-vine copulas, each edge E is equivalent to a bivariate copula. A R-vine tree on N variables (keys in Figure 7.2) consists of connected trees $T_1, \ldots T_{K-1}$ with nodes K_i and edges E_i for $i = 1, \ldots, K-1$, which satisfy the following conditions.

1. T_1 has nodes $K_1 = 1, \ldots, K$ and edges E_1.

2. For $i = 2, \ldots, K-1$ the tree T_i has nodes $K_i = E_{i-1}$.

3. Two edges in tree Ti are joined in tree T_{i+1} if they share a common node in tree T_i.

We restrain describing the R-vine copulas further as it is out of the scope of the paper and comprehensive explanations of R-vine copulas are available in Subsection 4.4.2 in (Jaworski, Durante, Hardle, & Rychlik, 2010).

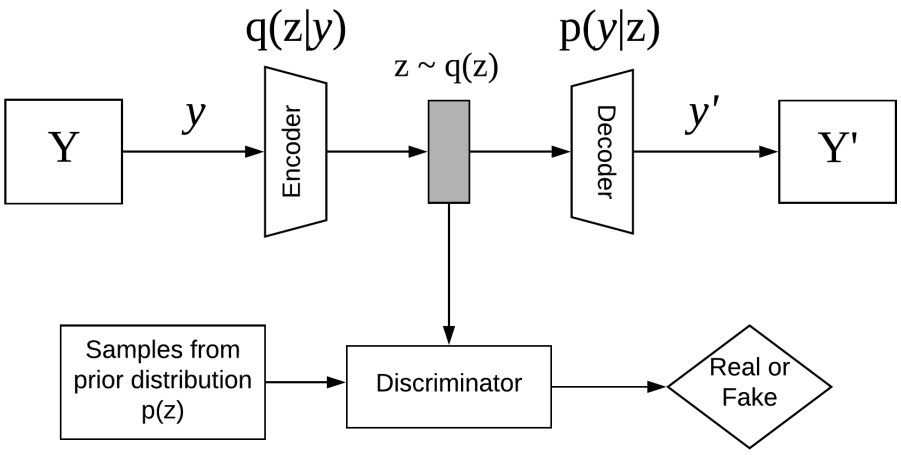


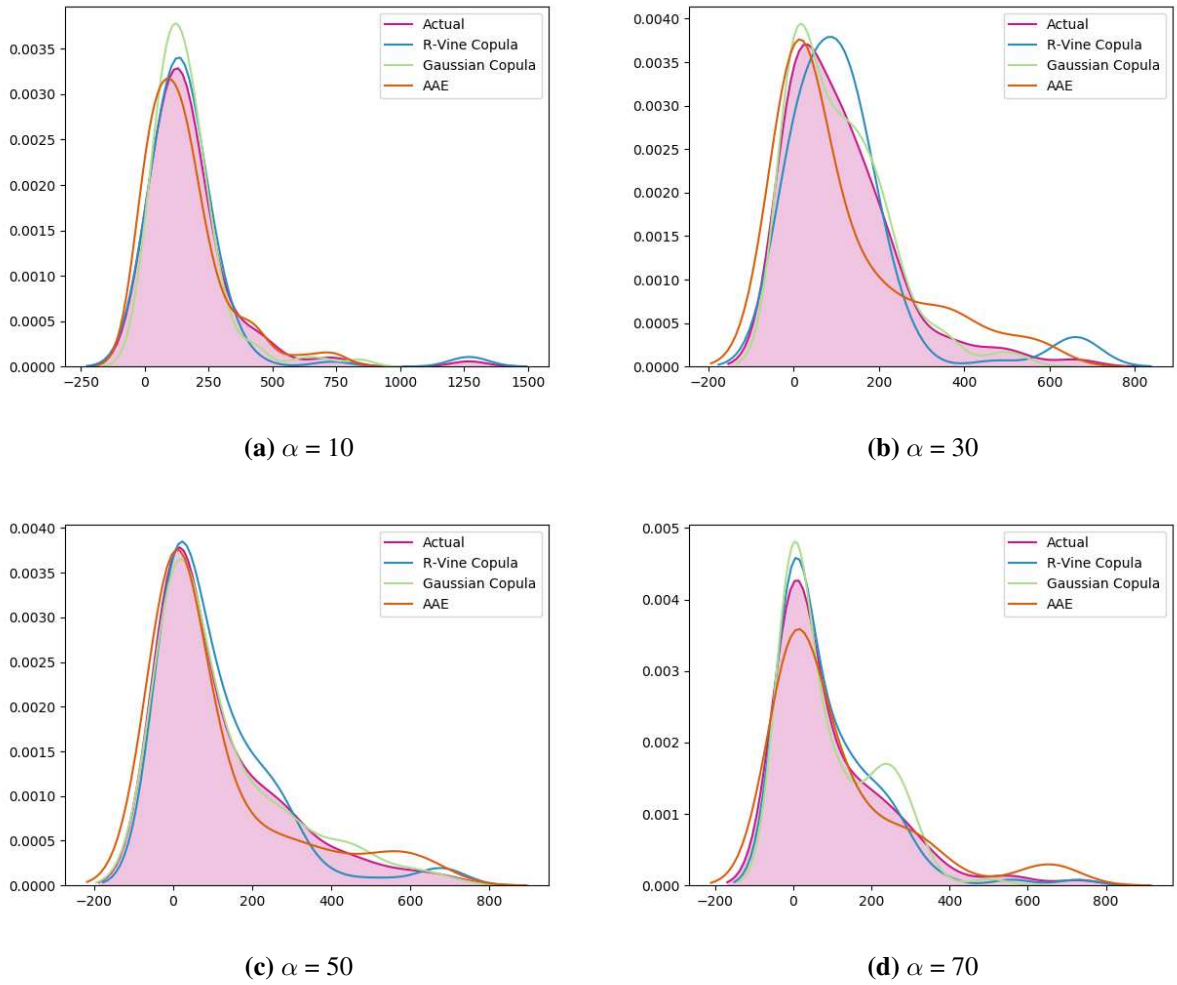
Figure 7.4: Architecture of the adversarial autoencoder

7.3.4 Deep generative approach: adversarial autoencoder (AAE)

The initial adversarial generative networks (GANs) (Goodfellow et al., 2014) only capture the continuous data distributions fluently (E. Choi et al., 2017), however various GAN approaches have been introduced for discrete data distributions recently. Adversarial Autoencoder (AAE) is one of the leading adversarial architectures for both continuous and discrete data generation. In AAE, the autoencoder forces a compressed knowledge representation of the actual input (either continuous or discrete), which reconstructs the same data distribution. In the training phase, the random set of objects in feature-mapped dataset (Y) is fed to the encoder in each iteration. In this work, the input of the AAE is defined as y, shown in Figure 7.4. In AAE, the input y reconstructs its data distribution (denoted as $p_d(y)$) from the latent code vector z. $q(z|y)$ and $p(y|z)$ stand for the encoding and decoding distributions respectively. $q(z)$ represents the aggregated posterior distribution of hidden code, which forms through the encoding function and the input data distribution. An aggregated posterior distribution $q(z)$ of the hidden code vector of an autoencoder for unconventional data can be defined as

$$q(z) = \int_y q(z|y)p_d(y)dy. \tag{7.7}$$

The basic principle of the AAE is that the autoencoder attempts to minimize the reconstruction error while the adversarial network tries to minimize the adversarial cost. Two simultaneous



(a) α = 10

(b) α = 30

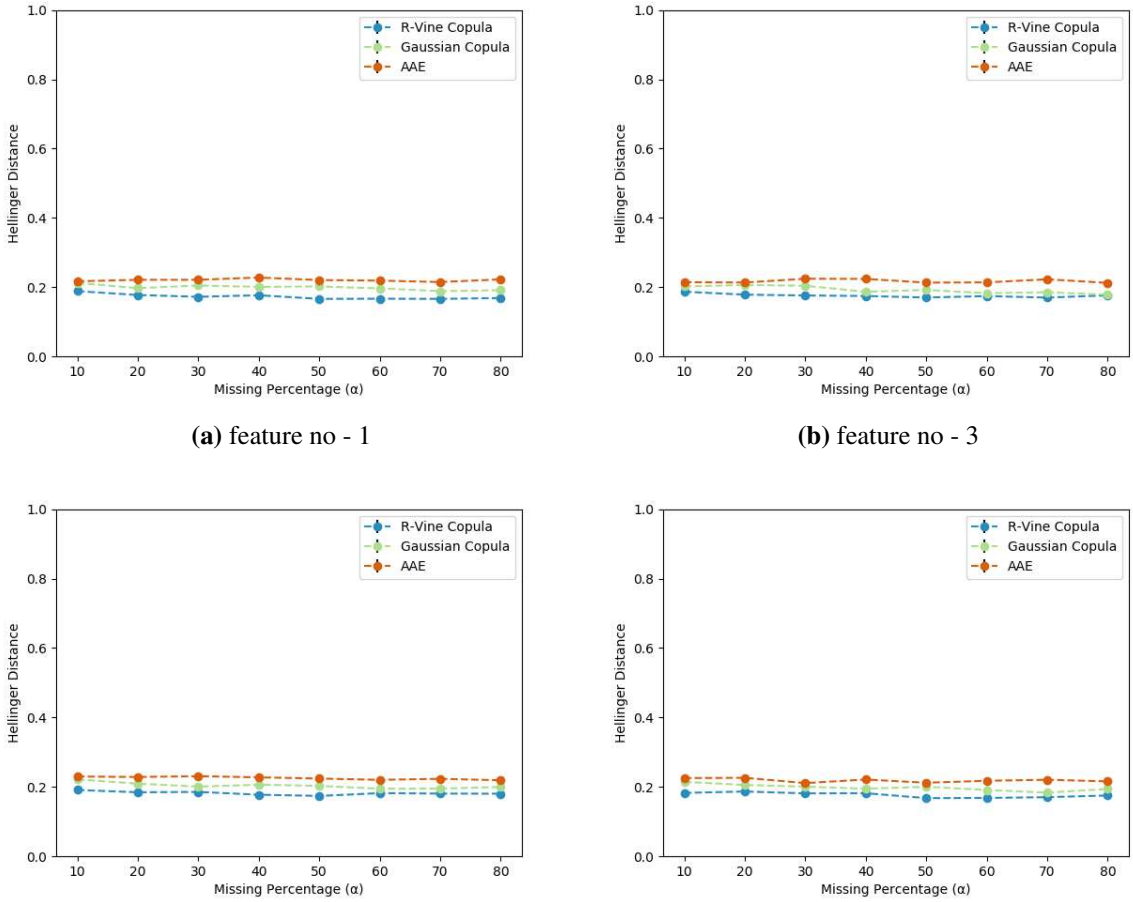
(c) α = 50

(d) α = 70

Figure 7.5: An example KDE (Kernel Density Estimation) plots of actual and generated data over several missing data percentages (α). The light red shaded section depicts the area of the actual distribution.

phases, *reconstruction phase* and *regularization phase* occur in each mini-batch during training. The reconstruction phase associates with the autoencoder of the network, and the network tries to minimize the data reconstruction error, often denoted as the loss. The regularization phase attaches to the adversarial component of the network where minimizes the adversarial cost to fool the discriminator by maximally regularizing an aggregated posterior distribution $q(z)$ to the prior $p(z)$ distribution.

After the training, the decoder defines a deep generative model that maps the prior distribution $p(z)$ to the data distribution $p_d(y)$ and generates a dataset Y' from the prior and decoding distribution as depicted in Figure 7.4.



(a) feature no - 1

(b) feature no - 3

(c) feature no - 6

(d) feature no - 9

Figure 7.6: Hellinger distance between actual and generated histograms for 4 features over missing data percentage (α). Y-axis and X-axis represent Hellinger distance and missing percentage respectively.

7.4 Simulation

We implement a `Python`[11] based simulator to generalize and validate our proposed feature-mapping technique for data generation using small UDS. In the simulation, we comparatively dis-

[11] https://www.python.org/

cuss the performance of both likelihood and deep generative approaches for UDS. For convenience, we use the same notations in Section 7.3. We use a multivariate random Gaussian distribution for data generation. For the generality of the value range, we apply the exponential of generated data as follows.

$$v_k = e^{\mathcal{N}(\mu,\Sigma)}$$

where μ and Σ are the given mean vector and the covariance matrix respectively. In the selection of covariance matrix Σ, we define a matrix $\mathcal{C}_{K\times K}$ where each element (c_{ij}) lies between 0 and 1 ($c_{ij} \in [0, 1]$) and K denotes the total number of features (keys). In this simulation, we use $K = 10$. Then, the covariance matrix is denoted as Σ where $\Sigma = \mathcal{C}\mathcal{C}^T$.

A random uniform distribution determines the unavailable values ($\delta = 0$) in the dataset. Let the percentage of unavailable values in a dataset be α. The α value is changed to generate different sets of unconventional datasets.

$$\alpha \in \{10, 20, \ldots, 70, 80\}$$

Suppose the random uniform distribution generates a value u ($\in [0, 100]$) for each data point, Then the multivariate dataset can be defined as follows.

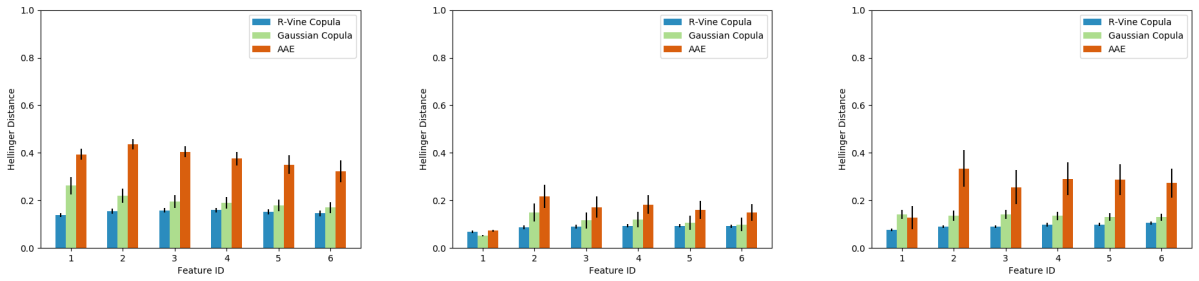
$$v_k = \begin{cases} v_k & u > \alpha \\ NA & else \end{cases} \tag{7.8}$$

Therefore, a generated ith object from the simulator o'_i can be similarly written as (7.1),

$$o'_i = \{(k, v_k, \delta_k), \forall k; k \in \mathcal{K}\}. \tag{7.9}$$

In this simulation, we generate 100 datasets for each unavailable data percentage (α). Each generated dataset has 100 records and it is considered to be a small dataset. As shown in Figure 7.2, we use the *feature-mapping encoder* to transform data to a structured format including the proposed

data approximation technique. In fact, we transform data in domain $\mathcal{X}$ to $\mathcal{Y}$ before applying to a data generation technique. As aforementioned, we use two multivariate copulas (Gaussian and R-Vine) and a deep generative network (Adversarial Autoencoder) to generate data in the simulation.



(a) Radicalization dataset **(b)** Stackoverflow dataset **(c)** Mimic III dataset

Figure 7.7: Hellinger distance between actual and generated histograms for first 6 features in the real datasets. The other features in stackoverflow and radicalization datasets also follow the same trend.

We use `Copulas`[12] Python library for Gaussian and R-Vine copulas for generating feature-mapped object data. In the deep generative approach, the AAE is configured for 500 epochs with 32 mini-batch sizes in the training phase. A `Keras Tensorflow` codebase[13] is used for the AAE implementation. Both the autoencoder and the discriminator are constructed with MLP (Multi Layer Perceptron) layers. In the AAE network, we configure the Adam algorithm as the optimizer with the learning rate of 0.0002 and the exponential decay rate for the 1st moment estimate (β_1) of 0.5. Further, we use the Mean Squared Error (MSE) as the loss function.

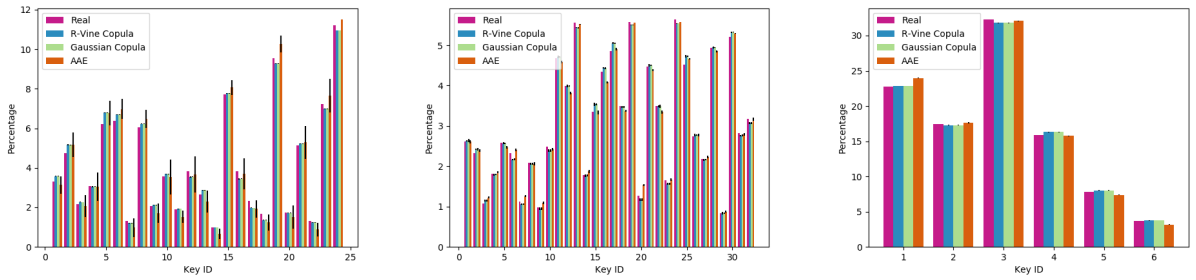
For each dataset, we generate a dataset consisting of 100 records (similar number of actual data records) using both copula and adversarial data generation techniques. Then, we transform the generated data (Y') to the original format ($\mathcal{Y} \rightarrow \mathcal{X}$). We perform various statistical evaluations to measure the similarity between actual (X) and generated data (X').

Figure 7.5 depicts the KDE (Kernel Density Estimation) plots of actual and generated data from all the generative approaches against different data missing percentages (α). The light red

[12]https://pypi.org/project/copulas/

[13]https://github.com/eriklindernoren/Keras-GAN

shaded section depicts the area of the actual distribution that helps to identify the actual distribution precisely. As shown in Figure 7.5, both copula and deep generative approaches almost fit into the actual distribution. We further calculate the difference between actual and generated data using their histograms as described in Subsection 7.4.1.



(a) Radicalization dataset **(b)** Stackoverflow dataset **(c)** Mimic III dataset

Figure 7.8: Probability of occurrences for actual and generated data in each feature. X axis represents all the features available in the real datasets.

7.4.1 Calculate the distance between two histograms

The difference between the actual and generated data in each technique is measured using the Hellinger distance (HD). Hellinger distance is a metric that quantifies the difference between two probability measures and widely used in various contexts due to the less computational intensity and has lower and upper bounds of 0 and 1, respectively (Sengar, Wang, Wijesekera, & Jajodia, 2008). We use 15 equal-sized bins to generate histograms for each dataset and then calculate the Hellinger distance.

Let $p(X)$ and $p(Y)$ are two probability distributions of X and Y datasets, respectively, on a finite sample space $\mathcal{S}$ where $p(X)$ and $p(Y)$ are with N-bins $\{x_1; x_2; ...; x_N\}$ and $\{y_1; y_2; ...; y_N\}$, respectively.

$$H(p(X), p(Y)) = \frac{1}{\sqrt{2}} ||\sqrt{p(X)} - \sqrt{p(Y)}||_2 \tag{7.10}$$

The HD satisfies the inequality $0 \leq H(p(X), p(Y)) \leq 1$ where 0 and 1 are the lower and upper bounds of the measure respectively. $H(p(X), p(Y)) = 0$ when $p(X) = p(Y)$ and disjoint

$p(X)$ and $p(Y)$ distributions represent $H(p(X), p(Y)) = 1$. For each dataset, we calculate the HD between the actual and generated histograms. Hellinger distances for 4 features (out of 10 features) are shown in Figure 7.6.

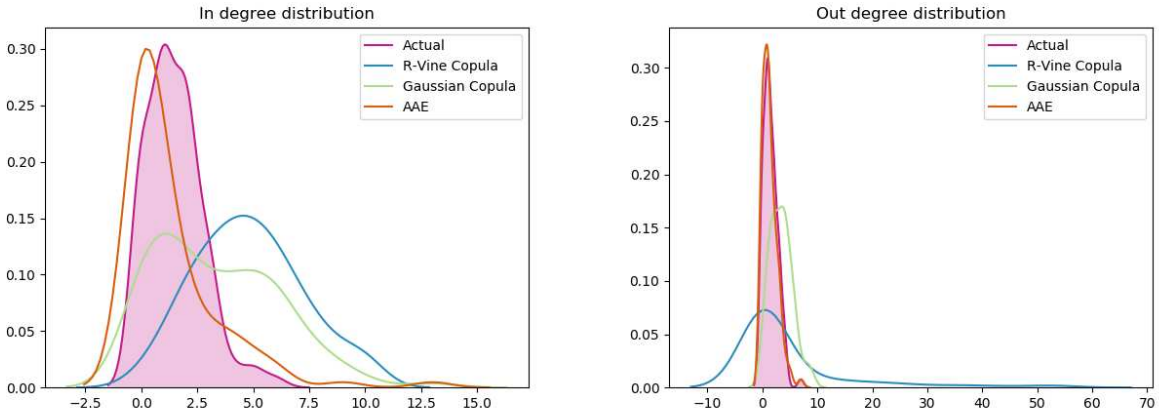
We claim that copula approaches have smaller HDs compared to the adversarial approach. Although all the Hellinger distances lie below 0.24 that implies the histograms of generated datasets are almost similar to the actual dataset. Further, our training dataset only includes 100 records which may insufficient to train a reliable deep-generative network. However, our feature-mapped techniques are still able to generate prominent datasets via adversarial autoencoders (AAEs) while overcoming all the challenges.

7.5 Computational details

We evaluate the performance of our proposed technique using three real-world datasets that represent a diverse range of domains. All the datasets are unconventional (UDS) with unavailable data points and small (a few hundreds of records). We have presented more details of the real-world datasets in Section 7.2.

Like in the simulation presented in Section 7.4, we calculate the Hellinger distances (HDs) between the histograms of actual and generated datasets. Figure 7.7a and 7.7b depicts the HDs of first 6 features in the radicalization and stackoverflow datasets respectively. We only select a few features to interpret because the rest of the features show the same trend too. Figure 7.7c shows the HDs of all 6 features. The HDs claim that the R-vine copula approach outperforms the Gaussian copula and AAE approach.

We further calculate the probabilities of occurrence for all the categories in three datasets. Figure 7.8 compares the percentages of categories in actual and generated datasets with each generative method. In fact, the probabilities of occurrences in 3 datasets claim that each generative method can generate almost similar percentages of features



(a) In-degree distribution

(b) Out-degree distribution

Figure 7.9: Probability of occurrences for actual and generated data in each feature. X axis represents all the features available in the real datasets.

7.5.1 Hierarchical data generation

We also experiment on generating the hierarchical structure of a network, which is beneficial in manipulating complex network topologies. In this work, we consider the adjacency matrix of a network with the exchangeability assumption where consider being a *graphon*, which is a symmetric measurable function (Janson, 2018). However, the availability of sufficient information in a network allows generating a complete hierarchical structure of the graph without the exchangeability assumption.

We generate adjacency matrices with the same techniques used to generate complex objects, and results are evaluated using the in-out degree distributions and the clustering coefficient of the network.

Table 7.1: Clustering coefficient of the networks over 100 generations in each technique

Method	Value
Actual	**0.018**
Vine-copula	0.334
Gaussian-copula	0.084
AAE	**0.024**

The average clustering coefficient of the generated networks over 100 runs are shown in Table 7.1. Figure 7.9a and 7.9b are depicted in and out-degree distribution of a single generation, respectively. Both evaluations (clustering coefficient and degree distributions) claim that the AAE-based generation outperforms the copula-based approaches. That further verifies that when the dataset's complexity increases (especially the data dimension increases), deep-learning approaches are more efficient and robust than likelihood approaches.

Therefore, we presented a novel feature mapping technique for UDS to statistically model such data, a demanding issue in social, behavioral disciplines, and many other domains. The proposed technique facilitated to apply both likelihood and deep-generative approaches make capable of generating high-fidelity UDS. We used Gaussian and R-vine copulas as the likelihood methods and an adversarial autoencoder as the deep-generative method for the data generation. We also presented a comparative analysis between likelihood and deep-generative approaches for UDS using multi-variate data simulation and three different real-world datasets.

We recognize the proposed data generation is applicable for many other research domains where the datasets are small and incomplete due to various data collection challenges. Our approach can assist in synthesizing data while maintaining the statistical characteristics of the actual datasets. Therefore, we apply this technique for radicalization trajectories generation that aid social scientists to expand their studies on significantly larger datasets. More details are presented in Chapter 8. Furthermore, we apply this data generation technique with modifications to a phishing detection application and video traces classification. Those details are discussed in Chapter 9 and 10, respectively.

Chapter 8

Radicalization trajectory generation

8.1 Introduction

Recent advances in data collection techniques allow collecting complex event data which form a heterogeneous set of events where an event (e_{ij}) defines a time of occurrence (t_i) and a category (c_j) separately. Therefore, such multi-category events not only concern about the time of occurrence but also the category of the event. Further, the multi-category events append extra dimensionality to the distribution which complicates the learning using existing technologies. In fact, multi-category events are greatly helpful to model the behavioral patterns of suspicious or specific individuals and groups in homeland security (Campedelli, Cruickshank, & Carley, 2019; Campedelli, Bartulovic, & Carley, 2019; B. W. Hung et al., 2018; B. W. Hung, Jayasumana, & Bandara, 2019), potential malicious network activities in cybersecurity (Peng et al., 2017), recommendation systems in consumer analytics (Vassøy et al., 2019), and the behavioral patterns of patients to determine certain illnesses (Islam et al., 2017; Mancini & Paganoni, 2019).

Major challenges especially in scenarios involving social analytics and privacy, are the limited data availability and the incompleteness of data due to data collection challenges such as data quality maintenance, data privacy, and confidentiality issues, but still a rigorous analysis is essential to produce accurate and reliable outcomes. A number of challenges limit the collection and access to data in many fields often resulting in small and incomplete datasets. Research and data collection on medical conditions, suicide, etc., have e.g., to strictly abide by privacy regulations (National Institutes of Health & Services, 2020). Scenarios involving social, political and crime behaviors are often incomplete due to data collection challenges such as data quality maintenance, privacy and confidentiality issues (National Institutes of Health & Services, 2020), but still a rigorous analysis with complete data is essential to produce accurate and reliable outcomes. So, there is a

critical need for a technique to capture and learn from the distribution, develop and apply machine learning algorithms, etc., for a small set of data some of which may be incomplete.

We use the data generation technique proposed in Chapter 7, which is capable of generating sparse, asynchronous, stochastic, and discrete events in continuous time based on a limited dataset. Adversarial training has recently evolved and can provide exceptional results in many data generation applications, mainly in image, audio, and video generation, while precisely mimicking the features of an actual dataset. In GAN architecture (Goodfellow et al., 2014), two neural networks, generator (G) and discriminator (D) compete together. D distinguishes the real and fake samples, and G confuses D as much as possible by convincing that generated samples come from the actual distribution. Ultimately, the generator will be capable of generating data samples that are similar to the actual samples. The primary GAN architecture (Goodfellow et al., 2014) only engages well for continuous and complete data distributions, and GANs have not been used for learning the distribution of discrete variables (E. Choi et al., 2017). Later, GAN architectures for discrete events were introduced (Makhzani et al., 2015; Yu, Zhang, Wang, & Yu, 2017) and also applied for trajectory generation using extensive training data (Xiao et al., 2018, 2017).

Adversarial autoencoders (AAE) are fluent in capturing latent discrete or continuous distributions (Makhzani et al., 2015). In this work, we use feature mapping modules that are present in Chapter 7 for accommodating incomplete data and make AAE capable of capturing data distributions of incomplete and small datasets. The incompleteness of the data points can occur in the following ways. The events have not been collected, or actors did not originally expose some events due to the dynamicity of these stochastic processes, which is the case especially in social and behavioral domains. The proposed method is capable of generating multi-category events on a large scale by leveraging relatively sparse, incomplete, and small datasets. Therefore, we utilize the proposed data generation technique for radicalization trajectory generation.

8.2 Radicalization trajectories

A radicalization trajectory represents a certain set of asynchronous stochastic discrete actions/events in continuous time (Upadhyay, De, & Rodriguez, 2018; Klausen et al., 2020) that shown in Figure 8.1. Each event binds with a behavioral indicator/category and time of the occurrence, representing a set of events related to an extremist.

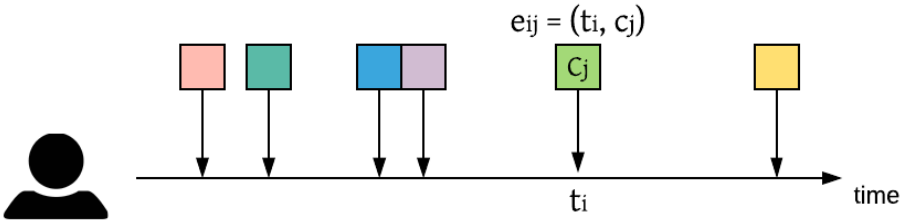


Figure 8.1: A basic representation of a radicalization trajectory with a set of multi-category events, each color represents a different category/behavioral indicator

A multi-category event in a radicalization trajectory is denoted as follows. An event e_{ij} is an **event** of category c_j occurring at time t_i as shown in Figure 8.1. Table 5.1 illustrate categories (i.e., c_j) in the dataset, where these heterogeneous events exhibits complex dependencies and correlations. Multi-category radicalization trajectories are described as follows. If the kth trajectory is $\mathcal{H}_k$ and its events are denoted as $e^k_{ij} \in \mathcal{H}_k$. Consider n events and m categories in the kth trajectory, then events are characterized as $e^k_{ij} = (t_i, c_j)$ where $i \in [1, n]$ and $j \in [1, m]$. Then,

$$\mathcal{H} = \{e^k_{ij} = (t_i, c_j) \in \mathcal{H}_k; k \in [1, N]\}, \tag{8.1}$$

where $\mathcal{H}$ represents N number of multi-category radicalization trajectories.

In some problem domains, i (time of occurrences) or j (categories) values may change rapidly and some categories may not be recorded frequently. More importantly, with many problems in domains such as social and behavioral sciences, not all events (e_{ij}) of a trajectory are known or observable due to the limitations of information gathering process, confidentiality constraints, unverifiability, deception, etc. A notable aspect of our work, is the use of sparse, incomplete, and

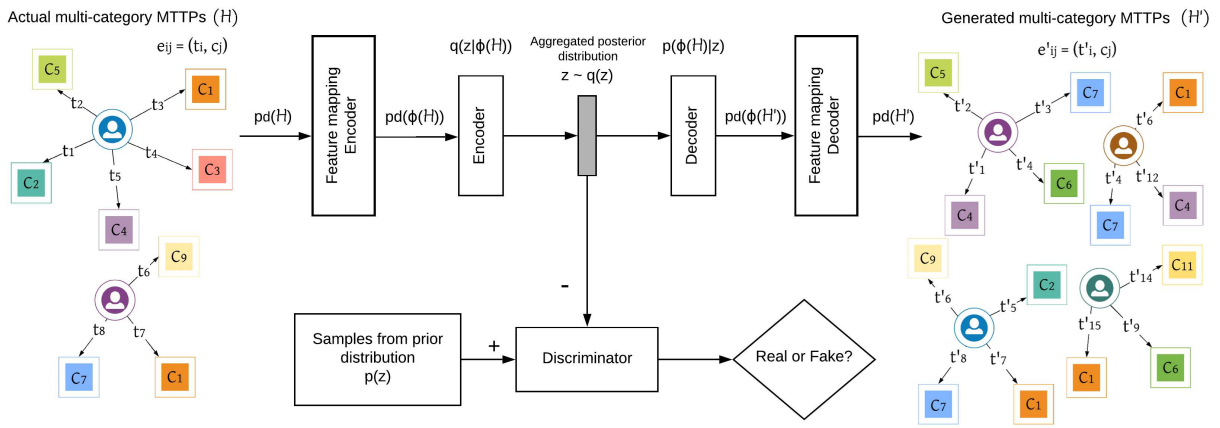


Figure 8.2: Architecture of an AAE for multi-category radicalization trajectory generation.

multi-category data where an actor(extremist) either did not carry out activities corresponding to a certain event categories and events or they carried them out, but such activities were not reported in the reliable or permissible sources. To address these challenges, we use a feature mapping encoder and decoder in Chapter 7, which are capable of capturing the sparseness and incompleteness of the data. The proposed feature mapping techniques consist of multiple steps, including the calculation of cumulative probabilities for each category and a data approximation technique for incomplete data (briefly described in Algorithm 5) w.r.t to the radicalization trajectory generation. More details of the feature mapping encoder and the decoder are discussed in Section 8.4.

8.2.1 Radicalization dataset

We evaluate the performance of our technique using 3 real-world datasets from a diverse range of domains. The Western Jihadism Database (Klausen et al., 2020) has almost all the incidents of terrorist actions committed in western countries including timestamps and yet it does not consist of unconfirmed or undiscovered activities that cause an incomplete dataset. We use 135 detailed pathways (multi-category events) of home-grown jihadists (Klausen, Libretti, et al., 2018) which have been extracted from radicalization trajectories of 335 known American jihadists (Klausen et al., 2016) as our real data distribution which covers over 24 behavioral indicators (categories).

8.3 Adversarial autoencoder (AAE) for radicalization trajectory generation

Figure 8.2 shows the adversarial autoencoder architecture for sparse, incomplete, and multi-category radicalization trajectories. The actual and generated trajectories are depicted in a tree structure where a root represents an actor of a trajectory. Square nodes stand for events, c_j denotes the jth category, and the value t_i on an edge (connects an actor and an event) denotes the ith time of occurrence. The general AAE only includes an autoencoder and a discriminator (Makhzani et al., 2015), but our proposed architecture in Chapter 7 consists of an additional feature mapping encoder and a decoder to capture characteristics of the sparseness and incompleteness of the data. The *feature mapping encoder* transforms multi-category trajectories ($\mathcal{H}$) to a cumulative distribution-based representation $\phi(\mathcal{H})$, which enables generating multi-category trajectories through an AAE. The *feature mapping decoder* is able to rearrange the generated multi-category trajectories $\phi(\mathcal{H}')$ to the actual format of multi-category trajectories ($\mathcal{H}'$).

The autoencoder forces a compressed knowledge representation of the original input which reconstructs the same data distribution. Initially, the original data distribution of multi-category trajectories $p_d(\mathcal{H})$ is fed into the *feature mapping encoder* which outputs the feature-mapped data distributions $p_d(\phi(\mathcal{H}))$. Then, the feature-mapped data is sent to the encoder where it compresses the data to a latent code vector z. $q(z|\phi(\mathcal{H}))$ and $p(\phi(\mathcal{H})|z)$ stand for the encoding and decoding distributions respectively. $q(z)$ represents the aggregated posterior distribution of hidden code which forms through the encoding function and the feature-mapped data distribution. An aggregated posterior distribution ($q(z)$) of the hidden code vector of an autoencoder for sparse, incomplete, and multi-category trajectories can be defined as

$$q(z) = \int_{\phi(\mathcal{H})} q(z|\phi(\mathcal{H}))p_d(\phi(\mathcal{H}))d\phi(\mathcal{H}). \tag{8.2}$$

The operating principle of the AAE is that the autoencoder attempts to minimize the reconstruction error while the adversarial network tries to minimize the adversarial cost. Two simultaneous

phases, *reconstruction phase* and *regularization phase* take place in each mini batch during training. The reconstruction phase relates to the autoencoder of the network, and it minimizes the data reconstruction error, often referred to as the loss. The regularization phase relates to the adversarial component of the network, where it minimizes the adversarial cost to fool the discriminator by maximally regularizing an aggregated posterior distribution $q(z)$ to the prior $p(z)$ distribution.

After the training process, the decoder defines a deep generative model that maps the prior distribution $p(z)$ to the feature-mapped data distribution $p_d(\phi(\mathcal{H}))$ and generates data samples $\phi(\mathcal{H}')$ from the prior and decoding distribution. The data generation can be interpreted as

$$p_d(\phi(\mathcal{H}')) = \int_z p(\phi(\mathcal{H})|z)p(z)dz, \quad \text{where } \phi(\mathcal{H}') \approx \phi(\mathcal{H}). \tag{8.3}$$

The generated feature mapped data ($\phi(\mathcal{H}')$) by the AAE is fed into the *feature mapping decoder* to transform the actual format of multi-category trajectories ($\mathcal{H}'$). Further details of the feature mapping encoder and the decoder are discussed in Section 8.4.

8.4 Feature mapping technique

The major challenge of applying the AAE framework to multi-category trajectories is that the data representation is structured; that is, each trajectory consists of a set of events belonging to various categories. In addition, each trajectory starts from an initial point (in the radicalization dataset, the date of birth of an actor) and continues with exposed multi-category events at different times. To implement the AAE framework discussed in the previous section, we propose a feature mapping which essentially maps the complicated data into the Euclidean space. This is achieved by a set of preprocessing steps including a data transformation for each category. The architecture of our proposed method is shown in Figure 8.2, and its functionality is summarized in Algorithm 5.

There are two major components: a feature mapping encoder (steps 1-3) and a decoder (steps 5-6) and the data generation (step 4). In **step 1**, all the event times (t_i) are transformed to the days (a_i) based on the initial point of each actor and shifted to a same days range. As a result, the

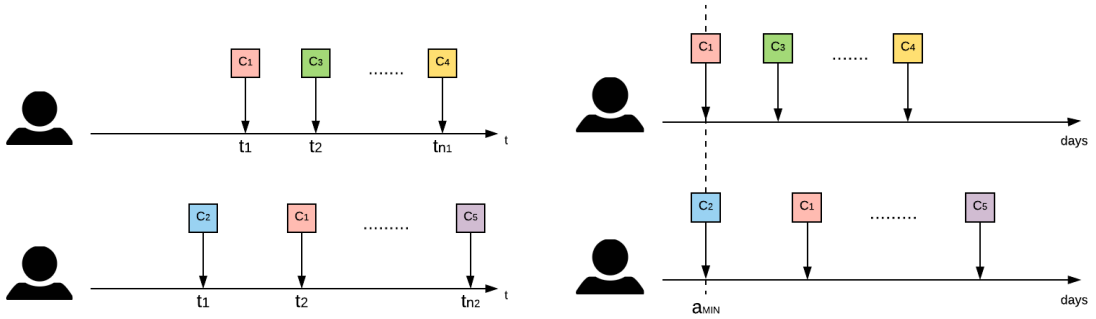


Figure 8.3: Radicalization trajectory visualization of two actors (extremists). The right-side image shows the trajacotories after shift to a minimum point (a_{min}.

Algorithm 5: Feature mapping & trajectory generation pipeline

---

Input: $\mathcal{H}$ (real trajectories), $e_{ij} = (t_i, c_j) \in \mathcal{H}$
Output: $\mathcal{H}'$ (generated trajectories), $e'_{ij} = (t'_i, c_j) \in \mathcal{H}'$
Step 1 Convert timestamps to days $e_{ij} = (a_i^{shifted}, c_j)$
Step 2 Replace with percentiles $e_{ij} = (P_{ij}, c_j)$
Step 3 Data approximation technique for incomplete marks
Step 4 Data generation via AAE $p_d(\phi(\mathcal{H})) \rightarrow p_d(\phi(\mathcal{H}'))$ where $e'_{ij} = (P'_{ij}, c_j) \in \mathcal{H}'$
Step 5 Replace percentiles with actual values $e'_{ij} = (a_i^{'shifted}, c_j)$
Step 6 Convert days to timestamps $e'_{ij} = (t'_i, c_j)$

---

updated events can be denoted as $e_{ij} = (a_i^{shifted}, c_j)$. For each trajectory, the value 0 is assigned to those categories that do not occur. As explained earlier, the absence of a certain category in an trajectory instance may be due to either that category not being associated with the MTPP, or limitations of data collection, or its presence not being recorded. The (inverse) percentile graphs for three categories (Convert date, Trauma, and Step towards violence) in the radicalization dataset are depicted in Figure 8.4. To measure the dissimilarity between two events in the same category, the difference of their shifted time is sufficient. However, such time difference varies a lot across different categories. In **step 2**, we propose to use the inverse percentiles instead. It helps to overcome the effect of sparseness and incompleteness of the data to a greater extent. For instance, the shifted time of the event e_{ij} can be transformed using $P_{ij} = F_{c_j}^{-1}(a_i^{shifted})$, where $F_{c_j}^{-1}$ is the inverse cumulative distribution function of the category c_j.

The inverse percentile distributions are shown in Figure 8.4 depict that there is a significance unavailability of events in categories. Therefore, unavailable events obtain higher percentile val-

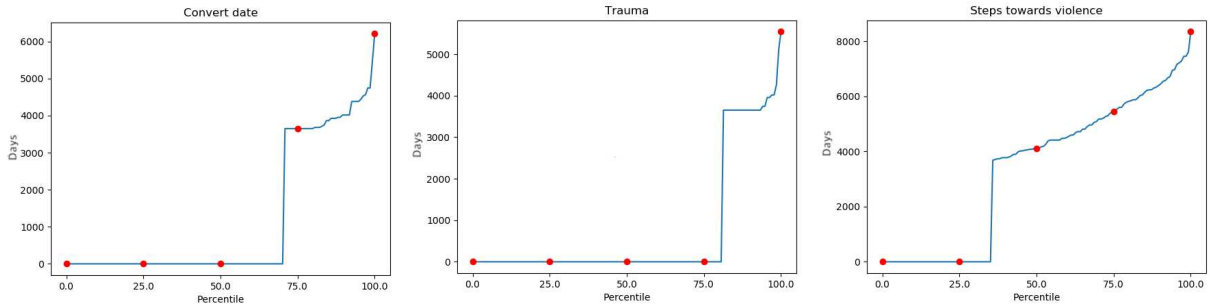


Figure 8.4: (Inverse) percentile graphs for categories in radicalization dataset ('Convert date', 'Trauma', and 'Step towards violence'). In this example, Y-axis represents days (a_i). Day 0 indicates the unavailability of events (eg: There are 70.37%, 80.74%, and 35.55% of unavailability of events for 'Convert date', 'Trauma', and 'Step towards violence' categories respectively).

ues which are sufficient to confuse the data distribution. In **step 3**, we use the *data approximation technique* that proposed in Chapter 7 to further mitigate the effect of the sparseness and the incompleteness of the data by changing the percentile values only for unavailable events (where $a_i^{shifted} = 0$) using a uniform distribution. $P_{c_j}(0)$ denotes the percentile value of unavailable events in jth category. The percentile values (P_{ij}) of unavailable events are substituted by randomly generated values (v_r where $v_r \in [0, P_{c_j}(0)]$) from the uniform distribution. Then, the percentile values are changed as follows. For a percentile value of jth category at ith time occurrence,

$$P_{ij} = \begin{cases} P_{ij} & \text{if } a_i^{shifted} \neq 0 \\ v_r \in [0, P_{c_j}(0)] & \text{otherwise} \end{cases} \tag{8.4}$$

After applying the *data approximation technique*, the updated events can be indicated as $e_{ij} = (P_{ij}, c_j)$. Steps 1-3 describe the sequential steps of pre-processing in the *feature mapping encoder* and produces feature mapped trajectories $\phi(\mathcal{H})$ as the input to the AAE (see Figure 8.2). The feature mapped trajectories can be denoted as follows. The kth feature mapped trajectory is $\phi(\mathcal{H}_k)$ and $e_{ij} \in \phi(\mathcal{H}_k)$, then

$$\phi(\mathcal{H}) = \{e_{ij} = (P_{ij}, c_j) \in \phi(\mathcal{H}_k); k \in [1, N]\} \tag{8.5}$$

where $\phi(\mathcal{H})$ is N number of feature mapped trajectories.

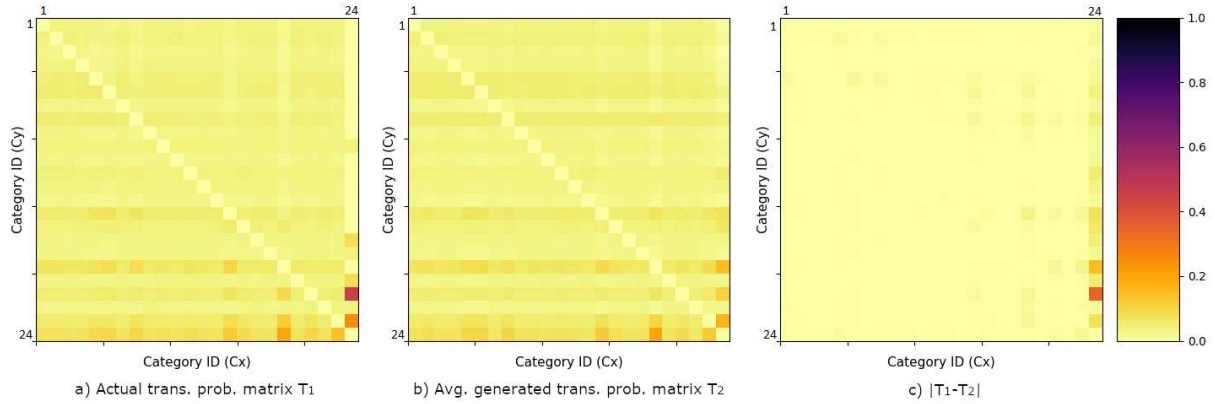


Figure 8.5: Conditional probability matrices for actual and generated datasets, and the difference.

In **step 4**, feature mapped trajectories $\phi(\mathcal{H})$ are fed to the AAE and data similar to the actual data is generated. The details of statistical methods that inspect the similarity between the datasets and the results are presented in the Section 8.5. Let M denotes the number of trajectories $\phi(\mathcal{H}')$, that are generated by the AAE. Then, the generated trajectories can be denoted as follows.

$$\phi(\mathcal{H}') = \{e'_{ij} = (P'_{ij}, c_j) \in \phi(\mathcal{H}'_k); k \in [1, M]\} \tag{8.6}$$

Then, the *feature mapping decoder* (see Figure 8.2) is applied to transform generated percentile values to the actual trajectory format. In **step 5**, a generated percentile value of an event of jth category at ith time occurrence P'_{ij} is converted to its day, based on the inverse percentile graphs in each category F_{c_j} (Figure 8.4), and reshifted to the actual days range. Then, an events is redefined as, $e_{ij} = (a'_i, c_j)$. In **step 6**, pre-defined or any distribution-based on the initial point can be utilized to convert days to real timestamps (t'_i) of the events. The advantage is that, it allows selecting any date range where provides flexibility to generate even future radicalization events based upon the requirement of the analysis. After the conversion, a generated events (e'_{ij}) of jth category at the ith occurrence t'_i is denoted as $e'_{ij} = (t'_i, c_j)$. The kth generated trajectory is $\mathcal{H}'_k$ and $e'_{ij} \in \mathcal{H}'_k$, we can define the generated trajectories ($\mathcal{H}'$) as

$$\mathcal{H}' = \{e'_{ij} = (t'_i, c_j) \in \mathcal{H}'_k; k \in [1, M]\}. \tag{8.7}$$

The introduced feature mapping techniques generate trajectories similar to the actual MTPPs. In Section 8.5, we show the similarity between generated and actual trajectories for radicalization using different statistical measurements. Further, we show the performance and robustness of the proposed technique compared to the baseline method (Markov-based generation), which was applied to the same dataset in a previous work (Klausen, Libretti, et al., 2018).

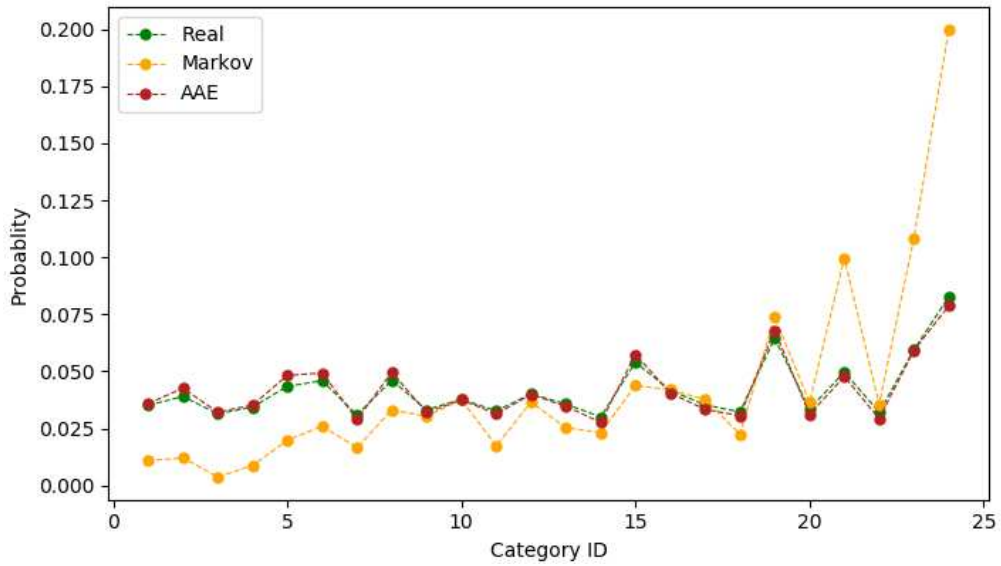
8.5 Results

In our experiments, the AAE is configured for 10K epochs with 32 mini-batch sizes in the training phase. The MTPP generation runs for 100 times and yields 10K MTPPs in each run. A `Keras Tensorflow` codebase[14] is used for the AAE implementation. Further, experimental details are available in Chapter 7. The typical machine learning approaches like Reinforcement Learning, RNNs, Wasserstein GANs require a significant amount of data to train a network. Therefore, such techniques are not applicable to our proposed approach. As the baseline, we compare the proposed data generation technique with a Markov chain approach which was applied to the same dataset in (Klausen, Libretti, et al., 2018). To compare with AAE-based generated trajectories, we produce datasets using their conditional probability diagrams of the Markov chain by running 100 times and obtain 10K pathways in each run. Conditional probability calculation is performed after applying the data approximation technique (step 3) described in Section 8.4. Here, the pre-processed data $\phi(\mathcal{H})$ is used to calculate actual conditional probability which provides further validation on our proposed feature mapping techniques. In the same way, generated data $\phi(\mathcal{H}')$ by AAE (before enter the feature-mapping decoder) is utilized to calculate generated conditional probabilities.

Figure 8.5 depicts the conditional probability matrices for actual data (panel (a)), generated data (panel (b)), and their difference (panel (c)) for the radicalization data using a color map. In each matrix, a cell (x, y) denotes the conditional probability $p(y|x)$ of category y given category x. Here, the probability is calculated based on the events $e_{ij} = (t_i, c_j)$. The probability difference matrix highlights the accuracy and the robustness of our proposed generation method by showing

[14]https://github.com/eriklindernoren/Keras-GAN

that the real and generated conditional probabilities are almost the same for every pair of categories. The only noticeable difference arises in Category 24 (max val = 0.35), which is due to the fact that 'date of criminal action' is the last event for almost all observations. The marginal distributions for radicalization and mimic data in Figure 8.6 further corroborates the similarity of the actual and the AAE-based generated datasets. The generated marginals are calculated based on over 100 runs.

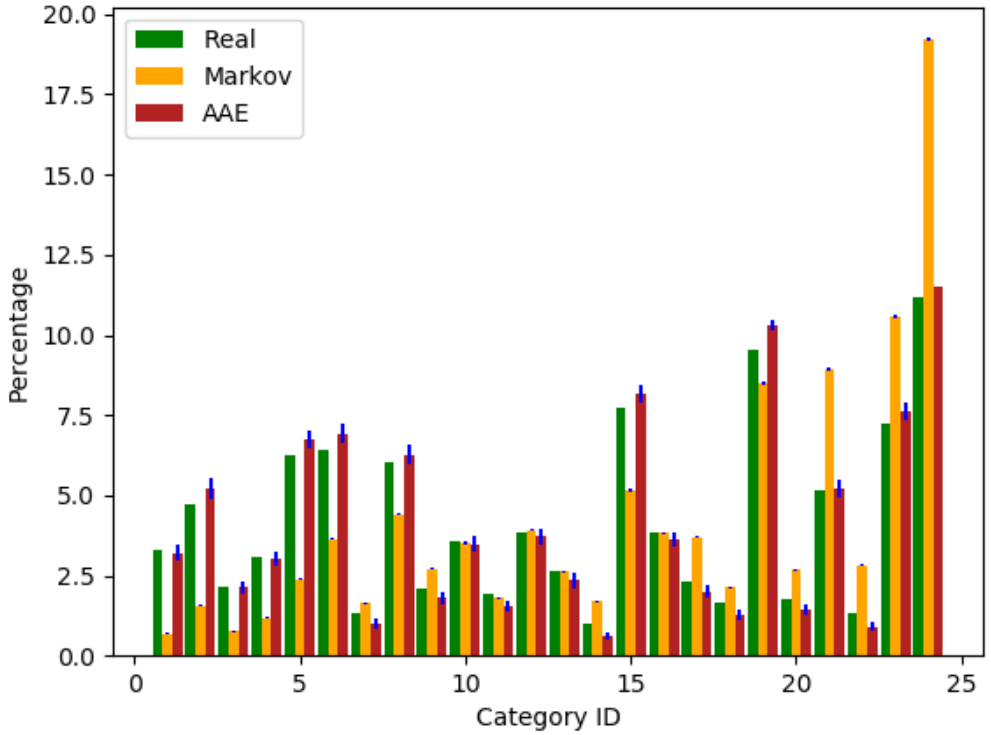


(a) Radicalization data

Figure 8.6: Marginal distribution for each category: column sum for the conditional probability matrix; Real (green), Markov generated (orange), and AAE generated (red) data.

We further calculate the probabilities of occurrence for the radicalization categories described in Table 5.1. Figure 8.7 compares the percentages of categories in actual and generated datasets. The error bar shows the standard error ($s.d./\sqrt{n}$ where $n = 100$) of each category and proves the robustness of the proposed multi-category MTPP generation technique.

We further compute the probability of occurrence for category pairs and triplets. The results for highly frequent pairs and triplets are shown in Tables 8.1 and 8.2, which further affirm the accuracy and the robustness of the data generation technique. For instance, the category pair [2,15] appears 36 (26.66%) observations in the real dataset, and in our generated data, it appears 2689 (26.89%) out of 10,000 generated samples. Consider two events denoted by $e_{i_1 j_1} = (t_{i_1}, c_{j_1})$



(a) Radicalization data

Figure 8.7: Bar plots for the probability of occurrence for each category based on real (green), Markov (orange), and AAE (red) generated data. Error bars (blue) represent the standard error over 100 runs.

and $e_{i_2 j_2} = (t_{i_2}, c_{j_2})$ where $1 \leq i_1, i_2 \leq n$ (n = total number of events in a trajectory) and $i_1 \leq i_2$, then the count ($\mathbb{V}_{j_1 j_2}$) is incremented by 1 for the $[c_{j_1}, c_{j_2}]$ pair. In other words, first-order transitions are calculated. We compute for all the pairs and few of the results are shown in Table 8.1 where it depicts transitions as portions of total number of trajectories (N), i.e., $\mathbb{V}_{j_1 j_2}/N$. The same method applies to triplets calculation in Table 8.2 where denotes three events $e_{i_1 j_1}$, $e_{i_2 j_2}$,and $e_{i_3 j_3}$, $1 \leq i_1, i_2, i_3 \leq n$ and $i_1 \leq i_2 \leq i_3$, then the count ($\mathbb{V}_{j_1 j_2 j_3}$) is incremented for the $[c_{j_1}, c_{j_2}, c_{j_3}]$ triplet and Table 8.2 shows $\mathbb{V}_{j_1 j_2 j_3}/N$ values for few triplets and further validates that second-order transitions are almost same for real and generated data.

8.6 Radicalization data evaluation

We complete the implementation of radicalization trajectory generation technique using both deep-learning and probabilistic approaches. The data collection of radicalization pathways is a

Table 8.1: Pairs (First-order transitions) comparison.

Pair	Actual	AAE Generated
[2,15]	36/135(26.66%)	2689/10000(26.89%)
[2,6]	26/135(19.26%)	1917/10000(19.17%)
[5,23]	48/135(35.56%)	3560/10000(35.6%)
[18,19]	8/135(5.92%)	559/10000(5.59%)
[5,24]	75/135(55.56%)	5607/10000(56.07%)
[2,19]	48/135(35.56%)	3552/10000(35.52%)
[10,8]	15/135(11.11%)	1179/10000(11.79%)
[6,23]	44/135(32.59%)	3387/10000(33.87%)

Table 8.2: Triplet (Second-order transitions) comparison.

Triplet	Actual	AAE Generated
[5,19,15]	14/135(10.37%)	1043/10000(10.43%)
[2,4,24]	11/135(8.14%)	796/10000(7.96%)
[3,21,24]	8/135(5.92%)	567/10000(5.67%)
[10,5,24]	11/135(8.14%)	867/10000(8.67%)
[1,8,23]	10/135(7.4%)	675/10000(6.75%)
[4,15,24]	18/135(13.33%)	1254/10000(12.54%)
[5,15,8]	9/135(6.67%)	739/10000(7.39%)
[21,23,24]	13/135(9.62%)	1065/10000(10.65%)

laborious task due to privacy concerns, dynamic and disparate data sources. Such challenges lead to having small and incomplete datasets. Researchers propose a novel synthetic data generation technique to mimic the behavior of extremists. This method is widely applicable in other domains as well, where the available datasets are small, sparse, or insufficient. Researchers implemented a novel feature mapping technique to overcome the data-specific challenges. As the deep-learning approach, researchers used an Adversarial Autoencoder, which is fluent in generating discrete distributions. Researchers already had experience with Adversarial Autoencoder that applied to other domains such as phishing websites detection and video traces classification.

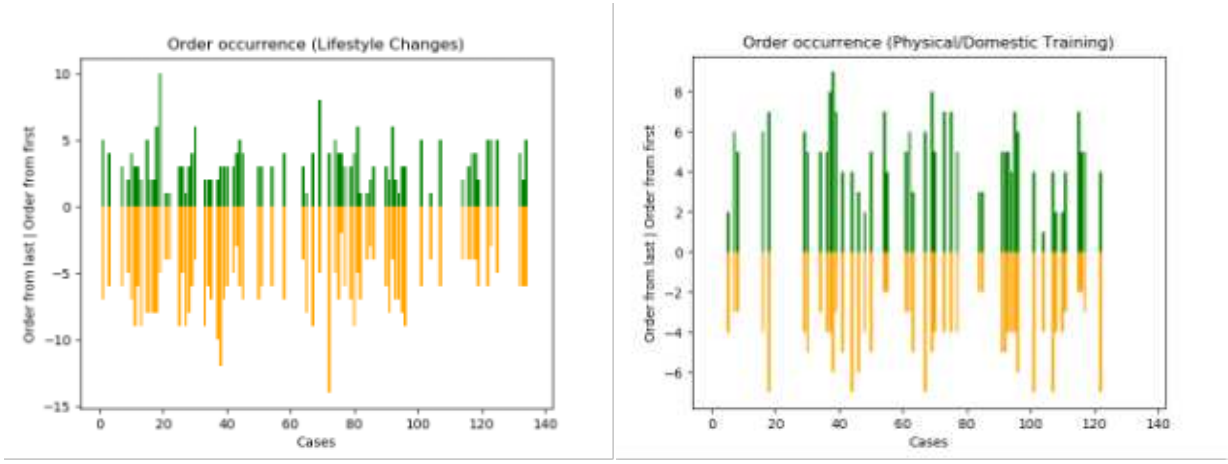
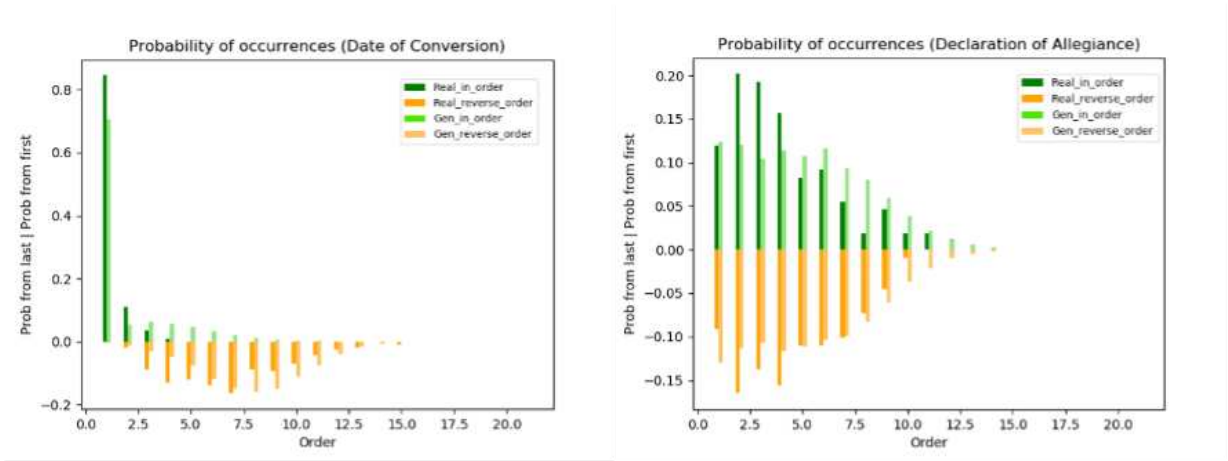


Figure 8.8: Order of occurrences plots on behavioral indicators: Lifestyle changes [left], Physical/Domestic training [right]



(a) Radicalization data

(b) Mimic data

Figure 8.9: Probability of occurrences (order from first and order from last). Date of conversion [left], Declaration of Allegiance [right]. Generated data is from the deep-generative based approach: Adversarial Autoencoder. x-axis represents the order of occurrence and y-axis depicts the probability from both the first indicator occurred (green) and from the last indicator occurred (orange). The probability based on the generated data via Adversarial Autoencoder depicts the light green (first indicator occurred) and light orange (from the last indicator occurred) bars. We generated plots for all the indicators for a complete evaluation.

The proposed data generation approach is beneficial to enhance the radicalization trajectory datasets to apply the latest machine learning techniques and other analytical models. It also helps to social science researchers to improve their coding process (manual data annotating process). Researchers also developed a method to evaluate the similarity between the actual and generated

trajectories. We calculate the order of occurrences that implies the chronological position of a particular indicator in each trajectory. The probability of order of occurrences provides insightful comparison as shown in Figure 8.9.

Researchers further expanded the evaluation of similarity between actual and generated data. We calculated the Euclidean distance between actual and generated plots in probability of occurrences for each indicator. The results are shown in Figure 8.10 and claim the Euclidean distance is less than 0.3 (first indicator occurred), which is prominent in terms of the order of occurrences among 20 indicators.

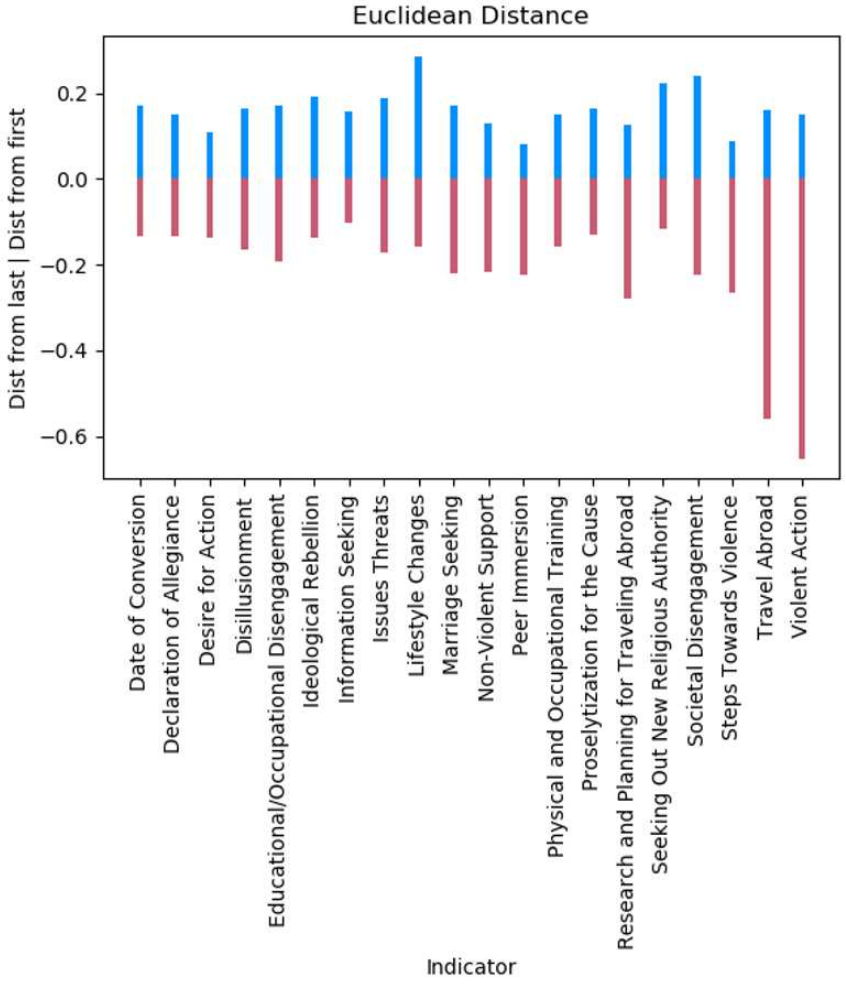


Figure 8.10: Euclidean distances between actual and generated plots in probability of occurrences for each indicator. Blue bars depict the distances from first indicator occurred and red bars show the distances from last indicator occurred.

Chapter 9

Phishing website generation

9.1 Introduction

Phishing attacks use social engineering and technology to steal user's identity (Dhamija, Tygar, & Hearst, 2006) and sensitive information (Singh, Sarje, & Misra, 2012). Commercial and government sectors have seen a proliferation of these attacks in recent years. Federal Bureau of Investigation (FBI) estimated 12.5 billion in financial losses worldwide from 78,617 reported incidents between October 2013 to May 2018 (Ho et al., 2019; Federal Bureau of Investigation (FBI), 2020a). Attackers mimic legitimate electronic communications and websites and seduce legitimate users into revealing their credentials. In this work, we focus on detecting phishing websites, which is one of the common phishing attacks. Researchers have been working on phishing website detection using diverse techniques and tools for many years. In this problem domain, supervised machine learning (Chen & Wang, 2020) appears to be a promising technique for phishing detection (Niakanlahiji et al., 2018; Sahingoz et al., 2019; Mao et al., 2019; Jain & Gupta, 2018). However, supervised techniques require labeled data in the training phase, which is sometimes tedious to collect and classify data. In the training phase, labels are assigned to the input data classifying them as legitimate or phishing. Then the trained model is then used to classify real-world data (Kirchner, Heberle, & Löwe, 2015) into genuine or phishing websites.

Furthermore, supervised machine learning algorithms need a sufficiently large labeled dataset for training classification models. The recent study shows that the existing phishing datasets have four issues of *availability*, *diversity*, *recency*, and *quality* (Dou, Khalil, Khreishah, Al-Fuqaha, & Guizani, 2017). Therefore, we focus on the problem of low-volume phishing datasets obtained from the real world. Collecting phishing data and labeling them is complicated and labor-intensive (Wong, Gatt, Stamatescu, & McDonnell, 2016). Gathering data in an adversarial context such as phishing poses unique challenges. In addition, researchers are often hesitant to share their

datasets related to cyber-security problems due to concerns such as privacy and liability. Only 10% of researchers shared their dataset publicly for a similar security networking problem (Abt & Baier, 2014). There are online repositories that collect the link to phishing websites like `PhishTank.com` or `OpenPhish.com`. However, such websites only provide a list of links. Obtaining the links, extracting the features, and converting them into a labeled dataset are complex tasks requiring significant expertise.

With the recent advancement of algorithms, adversaries actively try to bypass phishing detection algorithms or corrupt them. For example, an attacker can obfuscate code to invalidate the feature extraction process. Moreover, the low volume of existing phishing datasets (Dou et al., 2017) may cause the learning classifier not to converge and the performance to be inconsistent. In other words, the training model may be imperfect due to the absence of adequate data. In this work, we focus on how to increase the size of datasets while preserving the characteristics of existing data but without doing actual data collection. Such an approach is essential in phishing websites detection and any other domain when data is unavailable or when the data collection process is laborious and infeasible.

9.2 Proposed approach and key contributions

We define the goal of the attacker (adversary). We also discuss what the attacker can access and modify, especially with respect to the learning model, its parameters, and the dataset. We apply the adversarial autoencoder (AAE) network that described in Chapter 7 and 8 to mimic websites that are in-tune with the capabilities and characteristics of actual attackers. We investigate the similarity between synthesized and phishing samples at the *feature* and *instance* level to make sure the synthesized samples following same characteristics as real samples. This evaluates the validity of our generated synthesized samples via AAE.

In this work, we use four publicly available datasets developed by other researchers, and we train the learning models for each of those datasets. Our results are close to the results that have been reported by the authors of the datasets and validate the ability of our learning model to detect

the relation between features accurately and to discriminate phishing samples from legitimate ones. We then test if synthesized samples can circumvent the trained model and show that many new synthesized samples can bypass existing models. This shows that the learning models are prone to exploratory attacks (Huang, Joseph, Nelson, Rubinstein, & Tygar, 2011).

We show our proposed synthetic generation approach is useful on multiple purposes. First, it obviates the need for data collection where data may be unavailable or the data collection process may be infeasible for real-world attacks. Second, adding synthesized samples leads to a more precise classifier for phishing samples, which is a significant achievement because in many cybersecurity problems including phishing detection where data collection is a costly and tedious process. Our proposed approach can cut down the cost and time of data collection by having a hybrid method, including gathering a set of samples through the feature extraction process and generating new samples with an AAE. In addition, it may not be possible for many existing datasets to increase the number of samples through a data collection process due to practical difficulties like the unavailability of source code. Our proposed approach tackles these challenges and can generate new samples for those datasets and augment the dataset. Third, injecting these synthesized samples into training data generates more robust classifiers resistant to data poisoning attacks.

Our key contributions are as follows:

- As we demonstrate that the existing datasets lack a sufficient volume of samples for training phishing detection algorithms, we present an AAE (Adversarial Autoencoder) model to synthesize phishing data that mimic real phishing samples to enhance the training dataset.
- The proposed data generation method further widens opportunities in other research contexts that suffers from the lack of data for training models or rigorous analysis.
- We show that the actual samples and synthesized samples via AAE are similar enough in terms of two levels, i.e., feature and instance level. At the feature level, we use marginal distribution combined by calculating Euclidean distance. At the instance level, we clustered phishing samples and tested synthesized samples with that model.

9.2.1 Threat model

The attacker is modeled based on attackers' goal, knowledge, and influence in the context of phishing detection by a machine learning algorithm (Shirazi, Bezawada, Ray, & Anderson, 2019).

Attacker's Goal In the context of the phishing dilemma, we assume an attacker will attack the system's integrity by forcing the system to label a phishing instance as legitimate.

Attacker's Knowledge We assume an attacker only knows about features of the phishing instances but not the learning model parameters. This assumption is considered realistic as an attacker may have access to the definition of existing datasets but not the specific implementation. The adversary that has been modeled in (Shirazi et al., 2019) does not have any information about other system parameters like the algorithms that have been used, dataset instances, or learning parameters. The vulnerabilities of existing learning models against adversarial sampling attacks is shown in (Shirazi et al., 2019) However, in our current approach, we focus on synthesizing new phishing samples to address the existing limitations of dataset generation.

Attacker's Influence In phishing, there are mainly two types of attacks (Huang et al., 2011) : (a) *Causative Attacks* and (b) *Exploratory Attacks*.

In *Causative Attacks*, the attacker influences training data and can poison the training set with mislabeled samples to affect the training phase. The attacker is capable of poisoning a piece of data or whole dataset depending on the accessibility.

In *Exploratory Attacks*, the attacker aims the system's integrity in which the attempts are toward bypassing the learning strategy to exploit hidden spots in the learning model. In this attack, the attacker crafts intrusions so to avoid the classifier without direct influence. In this work, our goal is to design a system that is resilient against exploratory attacks.

9.3 Adversarial autoencoder (AAE) for phishing website data generation

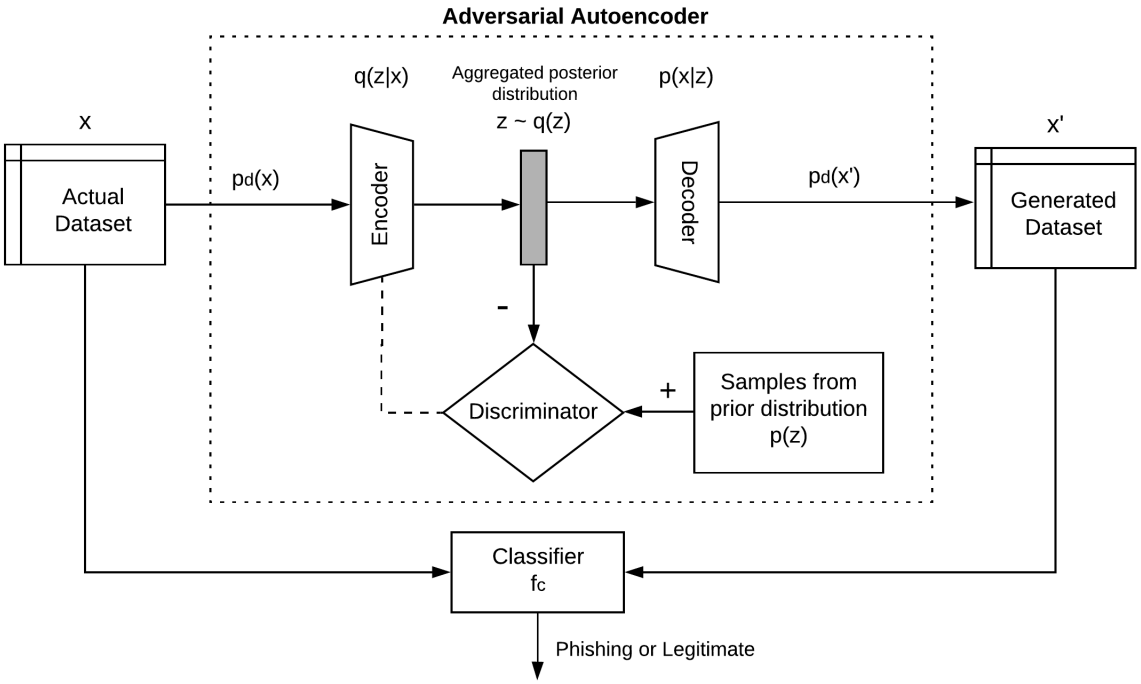


Figure 9.1: The architecture of our proposed approach. It consists of an adversarial autoencoder that generates synthesized data using phishing data. The top row depicts the ordinary autoencoder that reconstructs the data from the latent code z. The next row depicts the discriminative network that predicts whether the samples emerge from the hidden code of the autoencoder $q(z)$ or the user-defined prior distribution $p(z)$. $p_d(x)$ denotes the data distribution. $q(z|x)$ and $p(x|z)$ denote the encoding and decoding distributions respectively. After the data generation, a machine learning classifier (f_c) described is used to determine whether the synthesized samples belong to legitimate or phishing sites.

We use the adversarial autoencoder for synthesizing samples that mimic the phishing websites. The adversarial autoencoder is capable of generating both continuous and discrete data distributions. Therefore, AAE is a perfect fit for generating discrete feature sets in phishing samples. The architecture of the adversarial autoencoder is shown in Figure 9.1. The autoencoder derives

a compressed knowledge representation of the original input, which reconstructs the same data distribution.

$$q(z) = \int_x q(z|x)p_d(x)dx \tag{9.1}$$

An aggregated posterior distribution of $q(z)$ on the latent code is defined with the encoding function $q(z|x)$ and the data distribution $p_d(x)$ as shown in Eq. 9.1 where x denotes real phishing dataset.

The AAE's operating principle is that the autoencoder seeks to minimize the reconstruction error while the adversarial net- work attempts to minimize the adversarial cost. Reconstruction phase and regularization phase are two simultaneous phases that arise during training. In the reconstruction phase, the autoencoder's data reconstruction error is minimized, often referred to as the loss. The regularization phase relates to the adversarial component of the network. It minimizes the adversarial cost to fool the discriminator by maximally regu- larizing an aggregated posterior distribution q(z) to the prior $p(z)$ distribution.

The simultaneous training process allows the discriminative adversarial network into believing that the samples from hidden code $q(z)$ come from the prior distribution $p(z)$ (Makhzani et al., 2015). A normal distribution is exploited as the arbitrary previous $p(z)$ in this work. After the training process, the adversarial network synthesizes samples similar to the actual samples via the prior distribution $p(z)$.

We train generative models for each dataset because each has different sets of distinct features. The feature values are varied in many value ranges. Thus, the values are normalized between -1 and 1 before feeding to the encoder and are denormalized after data generation from the decoder.

At the end of this step, we will have two datasets: *Original Dataset*, which has been used to generate adversarial samples and a new *Synthesized Dataset* that consists of new synthesized phishing samples that mimic phishing websites.

We fed the model with only phishing samples so all of the synthesized samples are phishing. The synthesized dataset has the characteristics of phishing datasets generated by real-world attackers. We combine these two datasets to feed them into a classification algorithm that can distinguish

phishing samples from the legitimate ones. This classifier is unaware of whether the samples are synthesized, which means generated by adversarial, or real, which means we got them from an existing dataset. The instances are labeled as legitimate or phishing and classifier will predict them accordingly.

The samples that have been generated by that adversarial network will be injected into our training set with correct labels. The use of synthesized samples solves two purposes at the same time. First, we increase the dataset size and alleviate the problem of data unavailability and data collection. As explained, in a problem like phishing detection that is not easy to gather sufficient phishing cases, and existing datasets suffer from a lack of enough instances. Second, it helps to make the existing learning algorithm resilient against adversarial attacks.

9.3.1 Datasets

We use four publicly available phishing datasets on the Internet, and the details of these datasets are given below.

Dataset 1**: DS-1**: Shirazi *et al.*(Shirazi et al., 2018) published their unbiased phishing dataset in 2018. Each instance in this dataset has eight features, and all are related to the domain name of the websites.

Dataset 2**: DS-2**: Rami *et al.*(Mohammad, Thabtah, & McCluskey, 2012) created this dataset in 2012 and shared it with UCI machine learning repository (Dheeru & Karra Taniskidou, 2019). This set includes 30 features that are divided into five categories: *URL based*, *abnormal based*, *HTML-based*, *JavaScript based*, and *domain-name based* features.

Dataset 3**: DS-3**: In 2014, Abdelhamid *et al.*(Abdelhamid, Ayesh, & Thabtah, 2014) shared their dataset on UCI machine learning repository (Dheeru & Karra Taniskidou, 2019). The features include HTML content-based features and some features that require third-party services inquiries, such as DNS servers that perform domain-name age lookup.

Table 9.1: Number of instances, features, and portion of legitimate and phishing websites in each dataset

Dataset	Data shape (#)		Instances (%)	
	Size	Features	Legitimate	Phishing
DS-1	2210	7	44.71	55.29
DS-2	11055	30	55.69	44.31
DS-3	1250	9	43.84	56.16
DS-4	10000	48	50.0	50.0

Dataset 4**: DS-4**: This dataset is the most recent, from the year 2018, that is publicly available and has been created by Tan *et al.*(Tan, 2018) and was published on Mendeley [15] dataset library. This dataset includes 48 features, a combination of URL-based and HTML-based features.

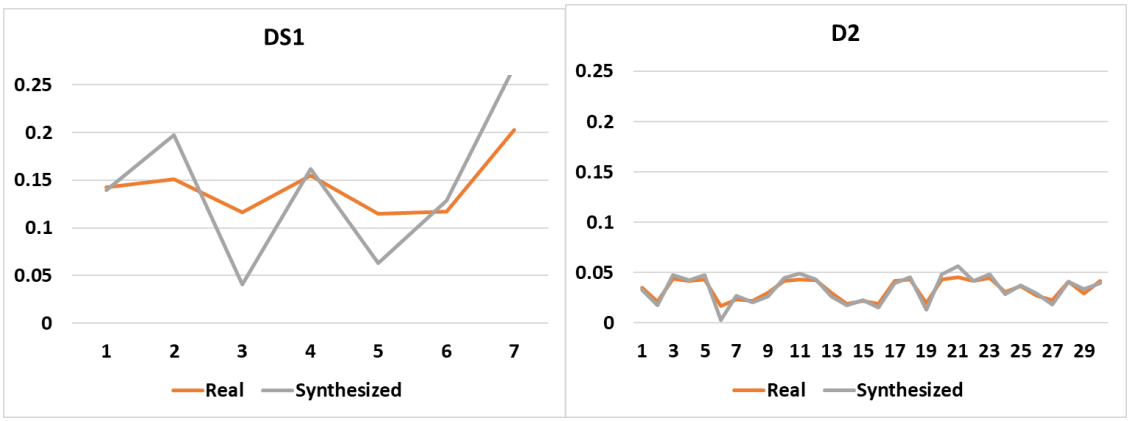
Table 9.1 summarizes the number of instances, features, and the portion of legitimate vs. phishing instances in each dataset.

9.4 Experiments and results

We demonstrate that the generated synthesized samples are similar to real phishing ones. We investigate the similarity at two levels: *feature* level and *instance* level. At the feature level, we need to ensure that the values assigned to the features in the synthesized samples are similar to real instances. We have done this through marginal distributions.

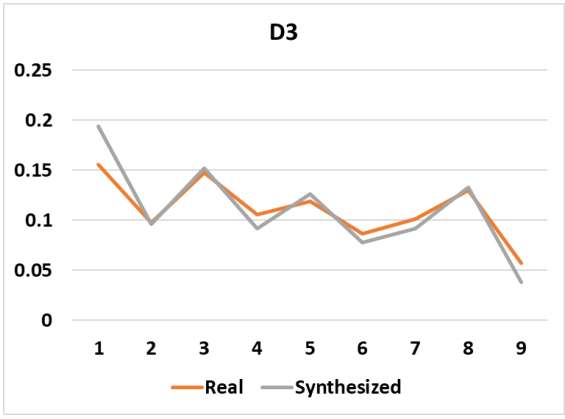
To calculate marginals, the transition probabilities were determined for normalized feature values (described in Subsection 9.3) to maintain the consistency across different ranges of values in features. The calculated marginal distributions are shown in Figure 9.2 and Figure 9.3 that further ensures the capability of the data generation technique. We also calculated the *Euclidean distance* between the marginal probabilities of the real and synthesized phishing data. The values are depicted in Table 9.2. The *Euclidean distances* are less than 0.13 across all the datasets which is very low. The minimum *Euclidean distance* of the column sum is 0.027 for the DS-4 dataset and the row sum is 0.025 for the DS-2 dataset. DS1 has the least number of features among datasets.

[15]https://data.mendeley.com/

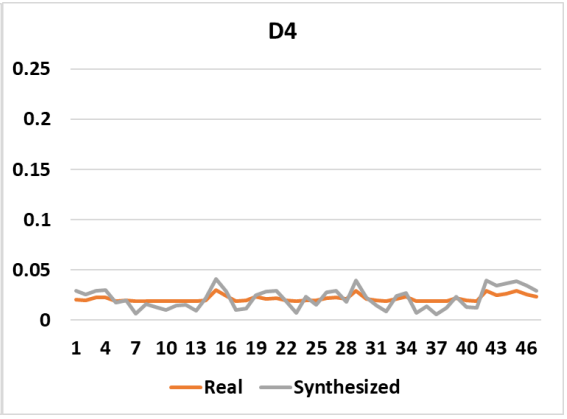


(a) DS1

(b) DS2



(c) DS3



(d) DS4

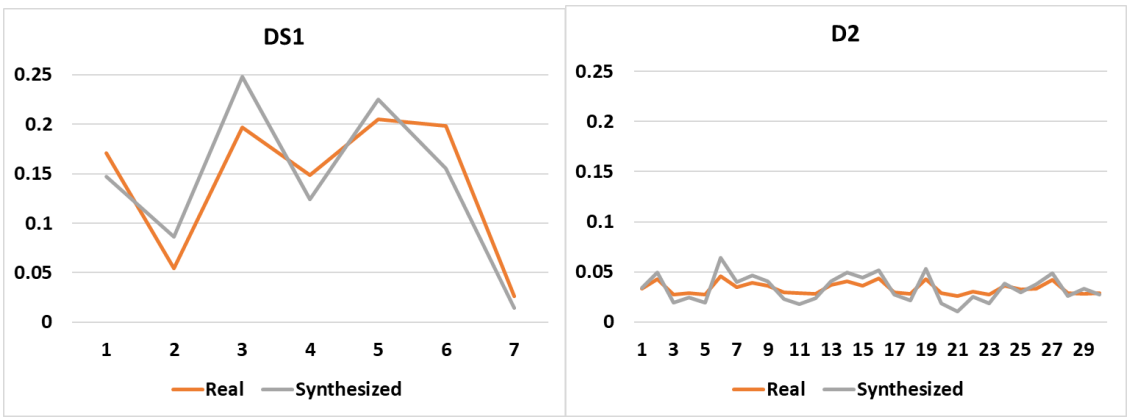
Figure 9.2: Marginal Distribution of X (Column sum)

We believe that is why the autoencoder slightly less performs on capturing the underlying multi-dimensional structure and transforming to the compressed latent code in DS-1.

After we checked the similarity of synthesized samples and real phishing samples at the feature level, we need to check the similarity at the instance level to make sure the final results of AAE are also similar. For this purpose, we created a clustering model based on real phishing samples and then tested on synthesized ones. We calculated what ratio of samples belongs to each cluster.

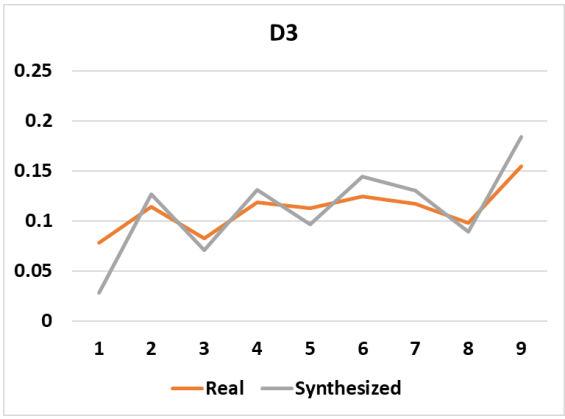
We used k-nearest neighbors (KNN) algorithm for clustering real phishing samples and used *elbow* method to get the optimum numbers of clusters for each dataset. The optimum number of clusters for DS-1, DS-3, and DS-4 is 6 clusters, and DS-2 is 8 clusters. Figure 9.4 shows the results of this experiment.

As Figures 9.3a, 9.3b, 9.3c, and 9.3d show, the ratio of synthesized samples is similar to ratio of real phishing samples and there is not any significant difference among them. This proves our

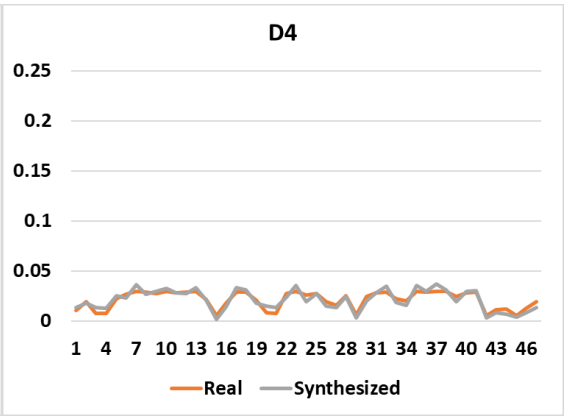


(a) DS1

(b) DS2

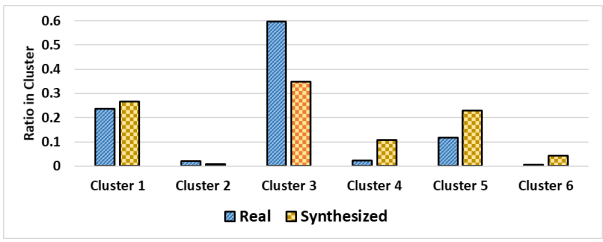


(c) DS3

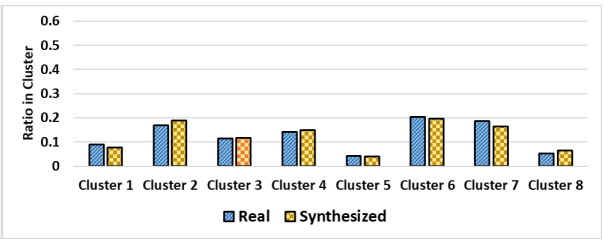


(d) DS4

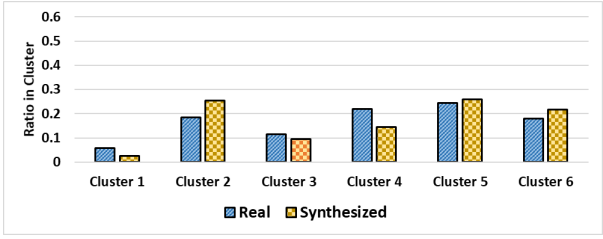
Figure 9.3: Marginal Distribution of Y (Row sum)



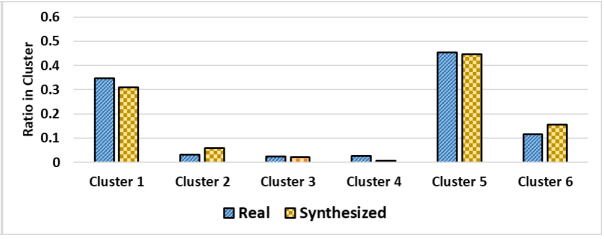
(a) DS1



(b) DS2



(c) DS3



(d) DS4

Figure 9.4: Ratio of real and synthesized phishing samples in each cluster in four datasets.

Table 9.2: *Euclidean distances* between real and synthesized phishing data.

Dataset	Euclidean distance	
	X (Column sum)	**Y (Row sum)**
DS-1	0.085	0.123
DS-2	0.042	**0.025**
DS-3	0.069	0.048
DS-4	**0.027**	0.050

adversarial generator can synthesized samples in the same distribution as the real ones. As we explained in Section 9.3, injecting synthesized samples may make the learning model resilient against exploratory attacks. To evaluate the second hypothesis, we injected 20% of synthesized samples into the training set, retrained the model, and called it *poisoned model*. We then tested this model with two types of samples: real and synthesized.

Supervised machine learning is a promising approach for phishing detection. However, sufficient volumes of data regarding phishing websites are unavailable and often infeasible to obtain. Towards this end, we demonstrated how Adversarial Autoencoders could be used to synthesize samples that mimic real phishing websites' data. We compared the similarity of the features and instances of the generated data to ensure that the attacker may realistically cause the generated data. Injecting synthesized data in the training set improved the accuracy and recall of the learning algorithms. Moreover, the learning algorithms that included some synthesized data also were significantly more robust to exploratory attacks. In this work, we validate our proposed data generation technique is beneficial in phishing website detection and improve the detection with synthesis samples.

Chapter 10

Video trace generation

10.1 Introduction

Video streaming has been dominating internet traffic for many years and is projected to consume up to around 80% of the global network traffic by 2022 (Cisco, 2019). Internet Service Providers (ISPs) and mobile network operators (MNOs) have struggled to manage the ever-increasing resource requirements for video streaming. With the recent emerging trends in Virtual Reality (VR) and Mixed Reality (MR), interactive videos are anticipated to strain the network resources even more than in the past decade. Unlike regular videos, interactive or $360°$ videos can demand 80 times more bandwidth for similar quality of experience (QoE) as conventional videos (Guan, Zheng, Zhang, Guo, & Jiang, 2019) and extremely low latency (less than 25ms) to circumvent motion or cyber-sickness (Liu et al., 2018).

ISPs and MNOs are adjusting to these challenges by increasing network capacity (Dimopoulos et al., 2016), deploying higher bandwidth technologies (Sukhmani, Sadeghi, Erol-Kantarci, & El Saddik, 2018), and operating traffic optimization techniques such as traffic shaping, balancing, caching, etc. (Dimopoulos et al., 2016). Traditionally, ISPs have relied on Deep Packet Inspection (DPI) to identify traffic flow characteristics and effectively deploy such techniques and new infrastructures. However, a broad adaptation of end-to-end encryption of traffic has significantly limited the effectiveness of transparent traffic optimizations that leverage DPI (F. Li, Chung, & Claypool, 2018). More than 75% of internet traffic is now end-to-end encrypted (Gutterman et al., 2019), while all popular video streaming services such as YouTube, Facebook, and Netflix only provide services over encrypted protocols like HTTPS. To this end, there has been a plethora of research in encrypted traffic classification leveraging recent advancements in machine learning (ML) (Y. Li et al., 2018; K. Choi et al., 2020; Dimopoulos et al., 2016; Schmitt et al., 2019). It has been shown that it is possible to identify which video is being transferred with simple data packet

features such as packet size and time (Y. Li et al., 2018; K. Choi et al., 2020). However, all these machine learning techniques require a significant amount of training data to train accurate classification models, which can go over a hundred thousands of samples (Dimopoulos et al., 2016) and are highly subject to the purpose of the analysis.

Video streaming data collection is challenging even in controlled experimental networks due to many restrictions. Some of them are mentioned as follows. Adaptive Bit Rate (ABR) algorithms in video streaming impose high variance in the time series of data packet features. End-to-end encryption mitigates third parties to easily filter specific traffic from particular service providers without sophisticated tools. Privacy and ethical issues prevent gathering data for training without proper user consent in live networks. Therefore, such difficult circumstances can result in either longer training time or partial, incomplete training data, leading to poor performance in ML classification.

Data synthesis using minimal raw data could be one approach to overcome these challenges in video traffic data collection. In addition to the probabilistic approaches such as copula (Ly et al., 2019), recent advancements in Generative Adversarial Network (GAN) based approaches have shown outstanding performance (Goodfellow et al., 2014) due to their capability of generating high-fidelity datasets in many contexts such as images (Karras et al., 2019), videos (Clark et al., 2019) and computer security applications (Shirazi et al., 2020). However, ordinary GAN architectures (Goodfellow et al., 2014) are only capable of capturing the distribution of continuous and complete data but cannot be used for learning the distribution of discrete variables in time series data (E. Choi et al., 2017). Therefore, many derivatives of GAN architectures were proposed, such as the Adversarial Autoencoder (AAE) (Makhzani et al., 2015) and Wasserstein GAN (Arjovsky et al., 2017) that are fluent in capturing both continuous and discrete data distributions. Nonetheless, neither of the above work focused on video traffic generation nor validated the performance of data generation with video traffic patterns.

In this work, we propose VideoTrain, a data generation framework for encrypted video traffic data realizing Wasserstein GAN-based data synthesis architecture with feature mapping tech-

niques. Video traffic generation is, in fact, more challenging than time-series data generations done in the past (Lin et al., 2019), as both temporal patterns and payload information of packets. We propose a novel percentile-based data mapping mechanism to encounter the high percentage of zeros in video traffic patterns. This allows us to generate different random snapshots of a single video trace which are used to train a Wasserstein GAN. We then investigate the effectiveness of synthetic data generation with a realistic use case—classification of 360° video traffic from normal video traffic. First, we synthesize traffic patterns for each video relying on the experimental data we collect and then investigate the classification performance by cumulatively adding them to the actual data. We have also developed classification models using Neural networks (NN) and traditional ML classifiers to achieve state-of-the-art classification performances.

We leverage more than 600 experimentally collected video traces, including 360° and normal videos from the two most popular providers YouTube (YT) and Facebook (FB). The results of our analysis show that VideoTrain data generation can improve the accuracy of our classifiers by 5 - 15%, increasing the accuracy from $\approx 85\%$ to almost 100% when adding nearly 450% of synthesized trace samples compared to the number of actual data. Given the length of video traces, this level of data synthesis saves nearly 98% training time compared to the time that would have taken if the data collection was to be done experimentally. We also observe that synthesized traces can maintain the exact characteristics of the actual traces both in their Kernel Density Estimation (KDE) and temporal domain. Moreover, the classification performance with only synthesized data has shown almost similar performance (more than 90% accuracy) compared to original and synthesized data. Our work shows the potential of developing privacy-aware data sharing mechanisms leveraging the VideoTrain framework.

10.2 Background and motivation

In this section, we further discuss challenges related to network data collection and the background of time series data generation using generative adversarial networks (GANs). It has been reported that there is a significant lack of public data access for video and audio applications (Ahmed

et al., 2018; D. Zhou, Yan, Fu, & Yao, 2018), especially one that involves representative samples of real users. There are a number of challenges involved in network data traffic collection, in particular video streaming traffic. One of the major challenges is associated with the significant effort and time required to develop data collection tools. The end-to-end encryption and opaque nature of traffic flows have made collecting network data passively through network middleboxes and endpoints a challenging task. As a result, significant effort and time are required to develop data collection tools and deploy them in real network systems. High complexity and dynamicity in mobile networks often create discrepancies in ground truth labeling, affecting its purity (Aceto, Ciuonzo, Montieri, & Pescapé, 2019). For example, Sheer volume of transactions and frequent lags when logging data for prolonged periods can negatively impact the purity of datasets, particularly in large MNOs and enterprise-level networks. In addition, privacy issues bound with network traffic prevents collecting data from users in the wild (D. Zhou et al., 2018; Aceto et al., 2019; Ganesan, Prashant, & Jhunjhunwala, 2012; N. Zhang et al., 2018). Exposure of sensitive information such as personal location and network usage details hinder potential customers to be participated in the experiments. These data collections often require ethical approval, which may require longer time to process depending on the network and country (Ganesan et al., 2012). Despite all the effort, poor geographical distribution of existing network infrastructure and failures in data loggers may either lead to longer experimental time or incomplete data (Ganesan et al., 2012; Z. Wang, Qian, Xu, Mao, & Zhang, 2011; Sherry et al., 2015).

The above facts conjecture that experimental data collection in communication networks has many challenges to address. Despite many hurdles, recent trends in machine learning networks such as Deep Neural Networks (DNNs) based approaches have demanded even more data for training than traditional ML models. On the other hand, data generation could be an effective alternative to get the required amount of data while reducing the time needed

10.2.1 Time-series data generation via generative adversarial networks (GANs)

GANs (Goodfellow et al., 2014) have become an alternative for data generation without extensive problem specific theoretical foundation or empirical verification (Yan, 2019). Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be the actual and generated distributions respectively. Generally in GANs, rather than estimating the density of the actual distribution ($\mathbb{P}_r$), we can define a random variable Z with a known (noise) distribution $p_z(z)$ and send it through a parametric function $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ that directly generates samples following a certain distribution $\mathbb{P}_\theta$ (Arjovsky et al., 2017). To achieve the above objective, two deep-neural networks—generator (G) and discriminator (D)—compete together in the training phase. D distinguishes the real and fake samples from $p_z(z)$ and G confuses D as much as possible by convincing that generated samples come from the real distribution ($\mathbb{P}_r$). Ultimately, the generator will be capable of generating data samples that are similar to real samples by mapping its distribution ($\mathbb{P}_g$) to $\mathbb{P}_\theta$. The competition between G and D is a two-player minimax game with value function V (G, D) (Goodfellow et al., 2014):

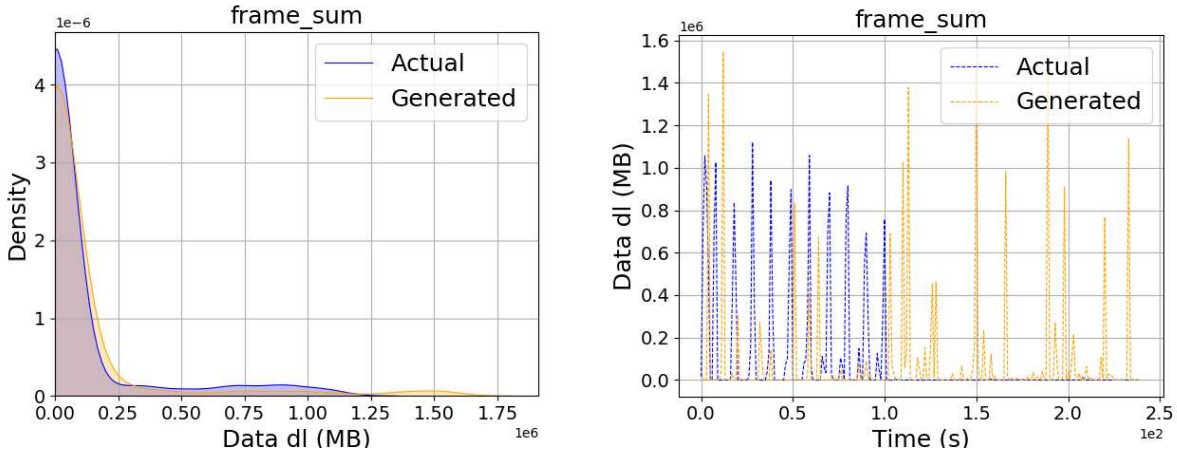
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_r(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{10.1}$$

where $D(x)$ is the probability calculated by the discriminator that x comes from $\mathbb{P}_r$ and G(z) is the generator mapping from the noise distribution to $\mathbb{P}_g$. The initial GAN architecture only engages well for continuous and complete data distributions, meaning that GANs are unable to learn the distribution of discrete variables efficiently (E. Choi et al., 2017). Hence, diverse GAN architecture for discrete events were later introduced (Makhzani et al., 2015; Arjovsky et al., 2017). Wasserstein GAN (WGAN) is a reliable approach in a wide range of domains: images (Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017) to internet traffic generation (Fathi-Kazerooni & Rojas-Cessa, 2020). WGAN has a unique loss function to calculate the difference between actual ($\mathbb{P}_r$) and generated ($\mathbb{P}_g$) data distributions compared to the ordinary GAN. It uses the Earth-Mover (EM) distance or Wasserstein-1 as the loss function (Arjovsky et al., 2017) while GAN calculates the loss via the standard cross-entropy (Goodfellow et al., 2014). In WGAN, the discriminator is

called the critic. Earth-Mover (EM) distance (Wasserstein-1) can be defined as follows.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{10.2}$$

where $\mathbb{P}_r$, $\mathbb{P}_g$ is the set of all joint distributions $\gamma(x, y)$, whose marginal distributions are $\mathbb{P}_r$ and $\mathbb{P}_g$. In other words, the similarity between actual and generated data is calculated by finding the infimum of the expected values of distances between data points from the distributions of actual and generated data (Fathi-Kazerooni & Rojas-Cessa, 2020). Further, training of WGANs does not require maintaining a careful balance of the D and the G, and also does not require a careful design of the network architecture. The Wasserstein loss function is also capable of providing a continuous and usable gradient compared to many other loss functions (Arjovsky et al., 2017).



(a) Kernel Density Estimation

(b) Time series with 0.5 bins

Figure 10.1: An example KDE and time series plots for a normal video trace that generated data using WGAN implementation with a min-max normalization. The results claim that it is capable of providing a reliable probability distribution but unable to preserve the temporal pattern.

However, the ordinary GAN implementations are unable to model the time series data with high fidelity because of the inability to maintain short and long-term correlations within the time series (Lin et al., 2019). To inspect this issue, we implement a WGAN with a min-max normalization method, proposed in (Fathi-Kazerooni & Rojas-Cessa, 2020) for video traces generation.

Fig. 10.1 depicts the frame sum with 0.5 bins for a normal video trace generated via the aforementioned method. While it claims that it is unable to capture the temporal effect of the video trace (Fig. 10.1b), it still generates an almost similar probability distribution (Fig. 10.1a).

Therefore, the hybrid models composed of Recurrent Neural Networks (RNNs) and GANs have been recently introduced to generate time series data while preserving short and long-term temporal correlations (Lin et al., 2019). These are comparatively complex deep neural networks that require a significant effort to train the models. However, with a simple data orientation, we can still utilize the naive GAN architecture to generate time series data with temporal effects. In (Brophy, Wang, & Ward, 2019), a time series trace is mapped to an image size of 64x64 pixels and a WGAN is utilized for data generation. The proposed method works well for periodical signals such as medical data; photoplethysmograph (PPG) and electrocardiograph (ECG).

But our problem is more complicated and deviated from the literature due to the following reasons.

1. video traces may not have periodicals patterns (even within the same video) and contains high fluctuations.

2. the zero temporal pattern often plays a significant role.

To address these challenges, we introduce a percentile-based data mapping technique and present VideoTrain in the next Section.

10.3 VideoTrain architecture

Initially, we briefly describe the overall workflow of VideoTrain followed by introducing the dataset used and the data synthesis process. After that, we explain our ML based traffic classification process which validates the effectiveness of the data synthesis process.

10.3.1 Overall VideoTrain workflow

Fig. 10.2 shows the main components of VideoTrain: *Data synthesis* and *Classification* phase. In the *Data synthesis* phase, we pre-process and synthesize new data. This dataset, further de-

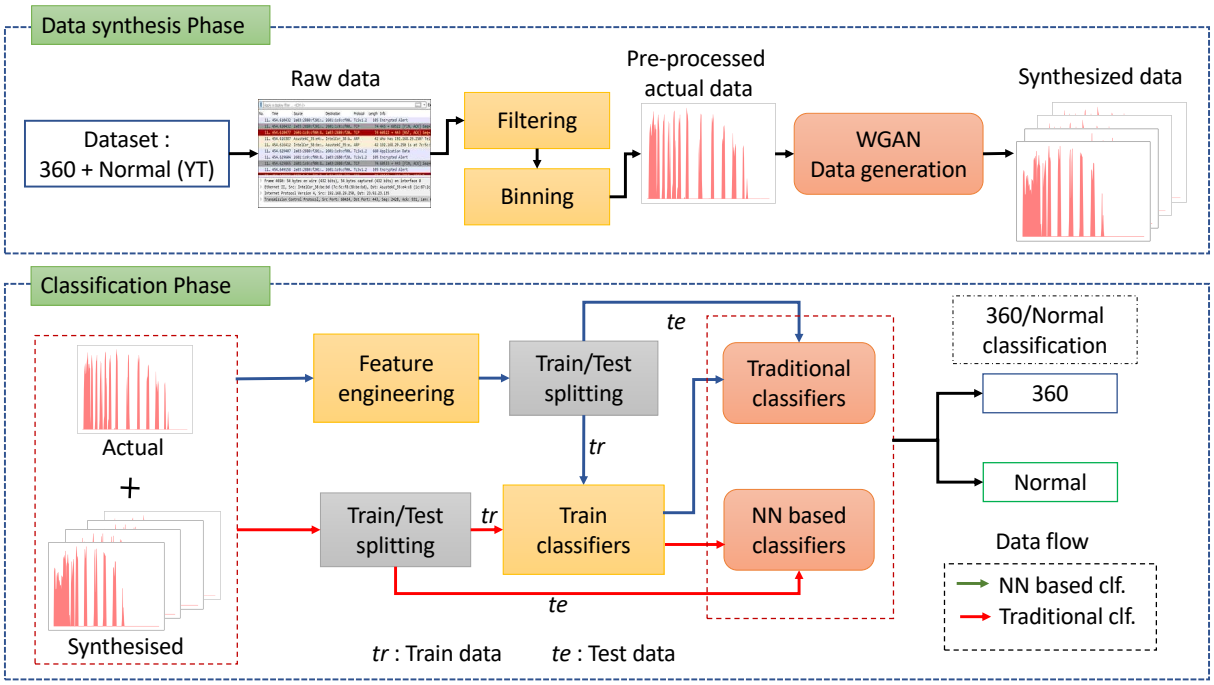


Figure 10.2: Overview of VideoTrain data generation framework.

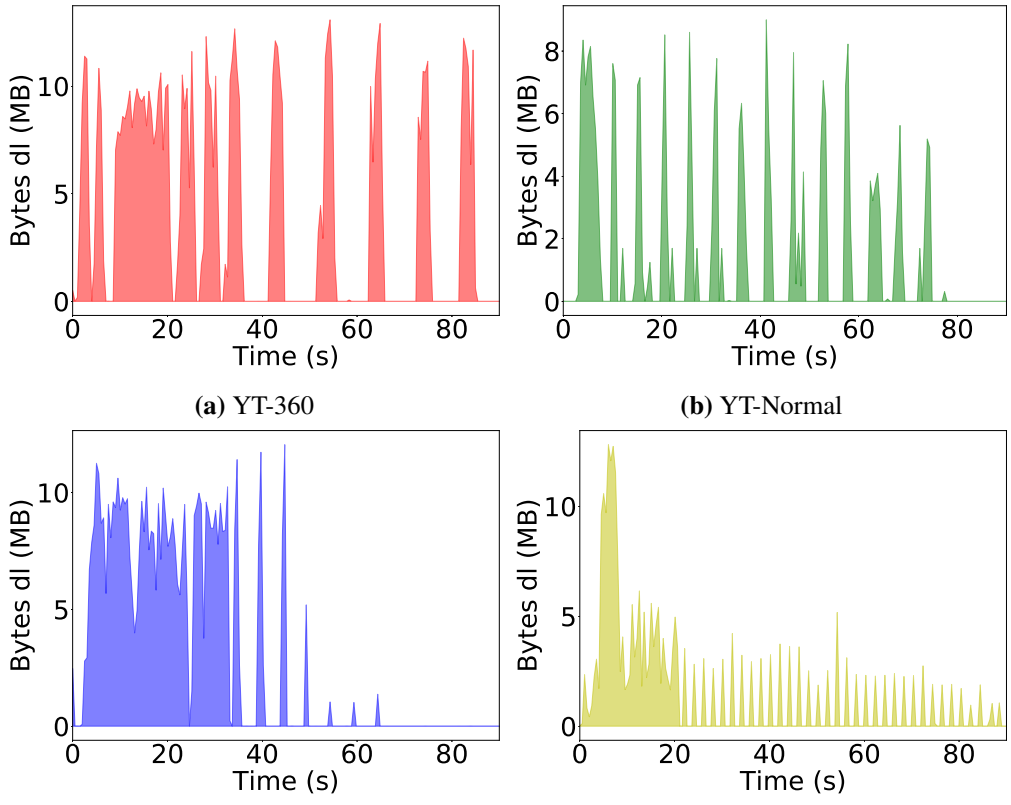
scribed in Section 10.3.2, represents a video streaming scenario containing 360° & normal video streaming data. Each video in the dataset is represented by a pcap trace. We extract bytes downloaded (bytes dl) data for non-overlapping bins for each trace. Finally, we input bytes dl time series pattern, trace by trace into our WGAN data generation model (Section 10.3.3) to synthesize data. For every trace we input, the model generates traces resembling the actual trace.

Classification phase combines actual and synthesized data in order to validate the synthesis process using both traditional (i.e. SVM, XGBoost) and NN based classifiers (i.e. MLP, CNN). To train the traditional classifiers, we generate overlapping bins of 5s with a step size of 1s for both actual and synthesized traces followed by deriving feature summary statistics over those bins for the bytes dl feature. For the NN based classifiers, bytes dl data is directly fed for training models. For both classifiers, we use 70/30 train/test split. Finally, the classifiers output whether the video traffic belongs to a 360° or normal category validating the effectiveness the data synthesis process.

10.3.2 Datasets

Table 10.1 summarizes our dataset comprised of YouTube (YT) and Facebook (FB) 360° and normal video traffic. The streaming videos were collected using two Android smartphones under non-controlled bandwidth conditions during three different times of the day: morning, afternoon and night. The videos have been streamed under many different genres such as documentary, sports, riding etc. which are commonly found in 360° video streaming (Afzal, Chen, & Ramakrishnan, 2017). We match normal videos to the same genres as well, to make the classification more robust to the content.

Fig. 10.3 shows some randomly selected row traces up to 90s duration from different video types which are sampled at 0.5s bins. We see noticeable differences in streaming patterns between different platforms mainly due to the proprietary streaming protocols and applications, different data collection devices and content of the video itself.



(a) YT-360 **(b)** YT-Normal

(c) FB-360 **(d)** FB-Normal

Figure 10.3: Downlink streaming patterns of selected traces of different video types for 90s duration

Table 10.1: Summary of the dataset

Platform	Content type	# of unique videos (video ids)	Total traces	Trace length (s)
YouTube	360°	50	160	120
	Normal	50	160	120
Facebook	360°	50	149	120
	Normal	50	147	120

10.3.3 Data synthesis phase

The overall work flow of *Data synthesis* phase is shown in Fig. 10.2-top. From the input pcap trace, we first extract the downlink traffic, followed by creating 0.5s non-overlapping bins of the packet data considering their timestamps. Then, we calculate *"Total bytes dl"* feature for each bin per trace. This process results in a time series data pattern, where each trace contains 240 bins with corresponding total bytes dl value.

Percentile-based data mapping

Let $\mathcal{K}$ denote the collection of all attributes in a dataset (a video trace) and x_k be an arbitrary value of the k^{th} attribute. Fig. 10.4 depicts an example percentile graph for an attribute and α_k denotes the percentile value when $x_k = 0$. In the initial step, we convert the values to its percentile values (x'_k) as follows.

$$x'_k = \begin{cases} F_k(x_k) & x_k > 1 \\ \alpha_k & x_k = 0 \end{cases} \tag{10.3}$$

where $F_k(.)$ is the cumulative distribution of k^{th} attribute and $x'_k \in [\alpha_k, 1]$. Here, we have a rigid threshold for zeros (α_k) which makes hard to train a deep neural network efficiently. As aforementioned, there are significant zeros temporal pattern in video traces and it is crucial to synthesize such patterns with high fidelity. To address this challenge, we map α_k to an uniform random distribution (u_k) where $u_k \sim [0, \alpha_k]$, disperses values evenly in the given range. Then, the

proposed data mapping can be defined as,

$$x'_k = \begin{cases} F_k(x_k) & x_k > 1 \\ u_k & x_k = 0. \end{cases} \tag{10.4}$$

Then, $x'_k \in [0, 1]$. Further, it allows to generate different random snapshots of a single video trace as shown in Fig. 10.5. These snapshots are used to train a WGAN. After data generation, we use the same Eq. 10.4 to re-transform the generated percentiles (y'_k) of k^{th} attribute to actual values (y_k).

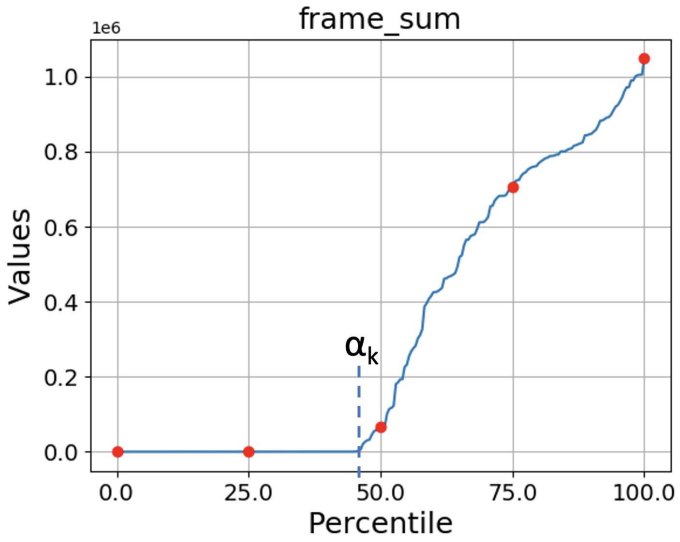


Figure 10.4: A (inverse) percentile graph for an attribute (frame sum)

10.3.4 Video traffic generation using WGAN

The details of the generator and critic structure of the WGAN are shown in Table 10.2 and 10.3, respectively. The generator and critic models in the WGAN are multi-layer perceptron (MLP) neural networks with three and two hidden layers, respectively, which is a comparatively simpler network design. The proposed technique in Subsection 10.3.3 suffers from overfitting (Lawrence

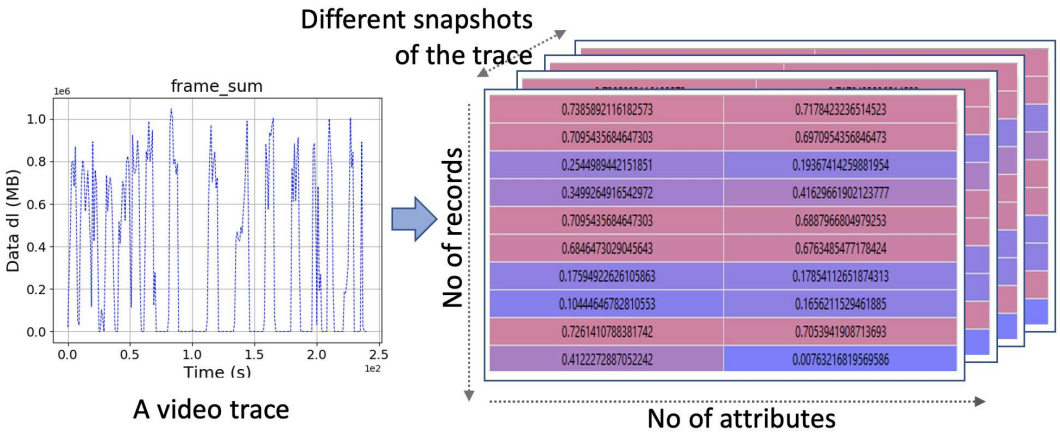


Figure 10.5: The data mapping pipeline for video traces generation

& Giles, 2000) and the dropout layers significantly reduce the issue and produce reliable results shown in Section 10.4.1.

Table 10.2: Generator architecture

Layer	Hyperparameter
Input	in_shape = (latent_dim,), act_func = 'leaky_relu'
Dense	h_units = 512, act_func = 'leaky_relu'
Dropout	drop_out_rate = 0.25
Dense	h_units = 512, act_func = 'leaky_relu'
Dropout	drop_out_rate = 0.25
Dense	h_units = 256, act_func = 'leaky_relu'
Dropout	drop_out_rate = 0.25
Output	h_units = (n_rec, n_feat, n_chann), act_func = 'tanh'

in_shape:input shape, h_units: number of hidden units number_of_estimators, act_func: activation function, max_depth

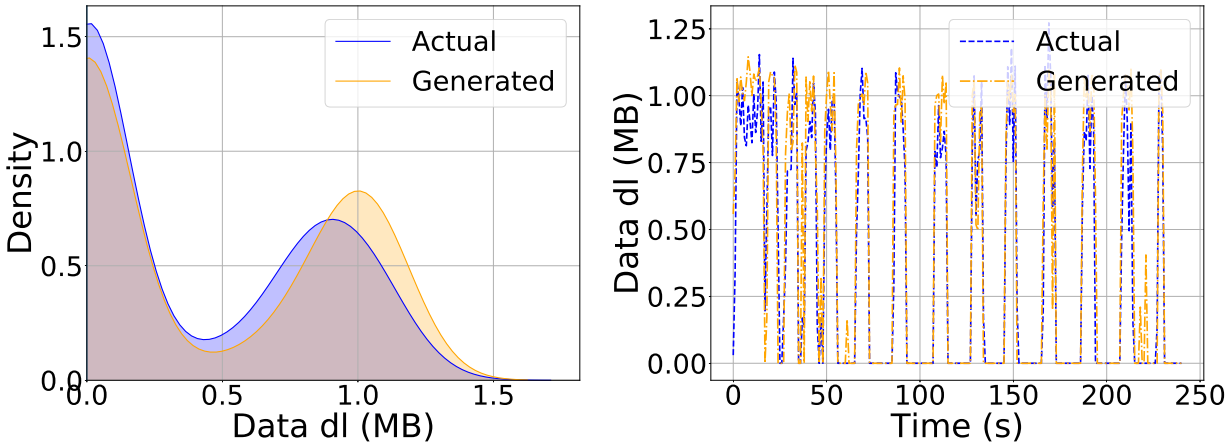
10.4 Results and evaluation

We first present statistical similarity of our generated data to the actual data validating WGAN based mechanism for time series video traffic feature generation. Then, we summarise our classification results whilst evaluating data synthesis process in different aspects. Finally, we discuss the possibility of privacy aware data sharing using only the synthesized data.

Table 10.3: Critic architecture

Layer	Hyperparameter
Input	in_shape = (n_rec, n_feat, n_chann)
Dense	h_units = 512, act_func = 'leaky_relu'
Dropout	drop_out_rate = 0.25
Dense	h_units = 256, act_func = 'leaky_relu'
Dropout	drop_out_rate = 0.25
Flatten	-
Output	shape = 1

10.4.1 Comparison of original and synthesized data

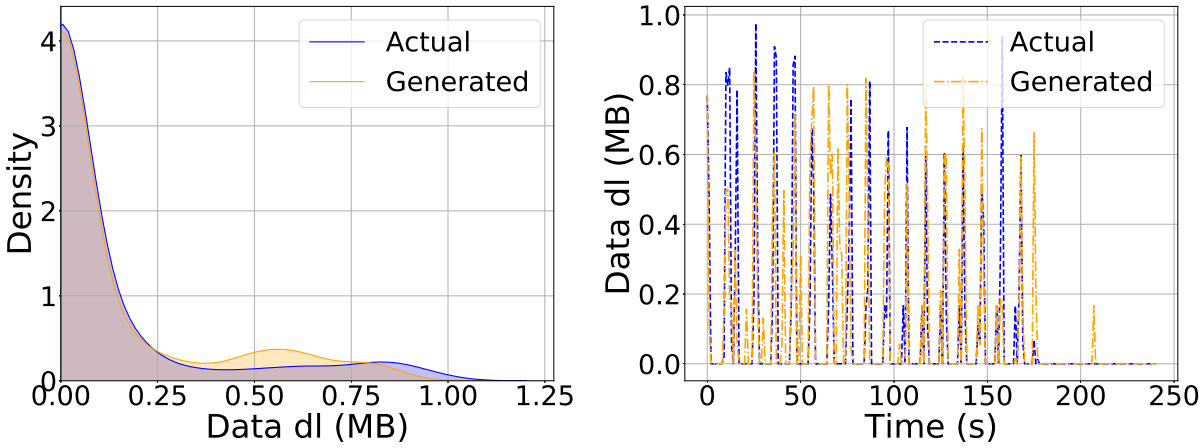


(a) Kernel Density Estimation (KDE)

(b) Temporal distribution

Figure 10.6: Comparison of actual and synthesized data (Bytes dl): YT-360°

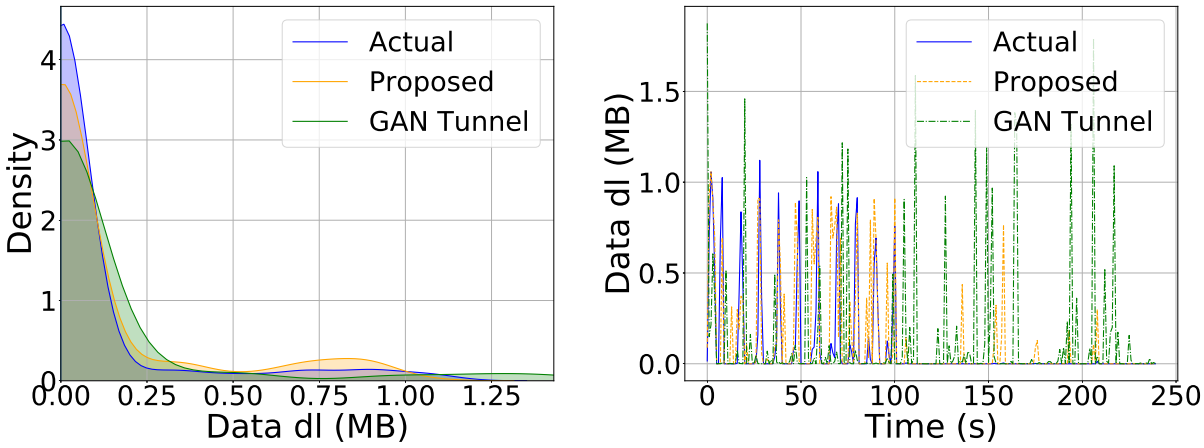
We compare the original and synthesized data based on KDE (kernel density estimation) diagrams and temporal distributions of bytes dl value, as shown in Fig. 10.6 and 10.7 for randomly selected YT 360° and normal videos, respectively. These graphs show that our proposed method is capable of maintaining the similarity of both probability and temporal distributions. We see that minimum and maximum values of actual and synthesized data are nearly the same in KDE plots (Fig. 10.6a and 10.7a) while the generated traces properly align with 0 value sequences and peaks (data chunks) of the actual traces in time domain (Fig. 10.6b and 10.7b).



(a) Kernel Density Estimation (KDE)

(b) Temporal distribution

Figure 10.7: Comparison of actual and synthesized data (Bytes dl): YT-Normal

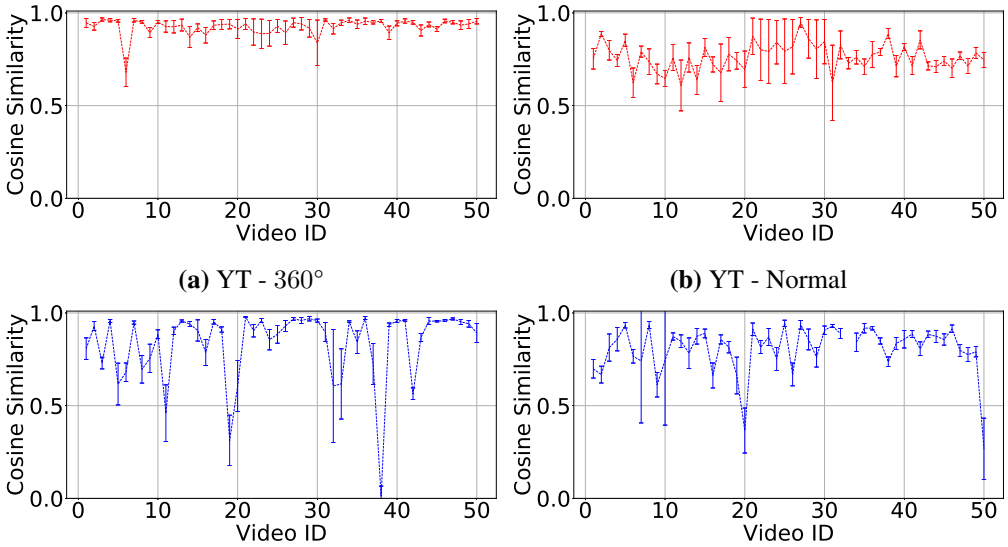


(a) Kernel Density Estimation (KDE)

(b) Temporal distribution

Figure 10.8: Comparison of proposed method and GAN Tunnel along with actual data (Bytes dl): randomly selected trace from YT-Normal

We compare VideoTrain with min-max normalization based WGAN data generation method, *GAN Tunnel* (Fathi-Kazerooni & Rojas-Cessa, 2020), to show that, it can overcome the problem of maintaining the temporal characteristics of time series data by ordinary WGAN (cf. Section 10.2.1). Fig. 10.8 shows the KDE and temporal distribution diagrams of a randomly selected YT-Normal trace. It highlights that, when compared to *GAN Tunnel*, peaks and zero-sequences of the trace from VideoTrain well align with the actual trace—particularly at the beginning of the



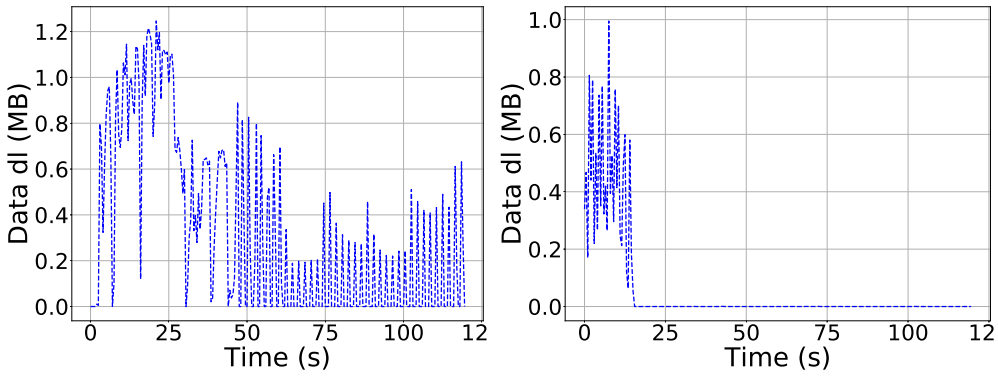
(a) YT - 360°

(b) YT - Normal

(c) FB - 360°

(d) FB - Normal

Figure 10.9: Cosine similarity between Actual and Synthesized traces by Video ID



(a) FB-Normal high cosine similarity

(b) FB-Normal low cosine similarity

Figure 10.10: Distribution of selected two actual traces from Facebook Normal category with high and low cosine similarity

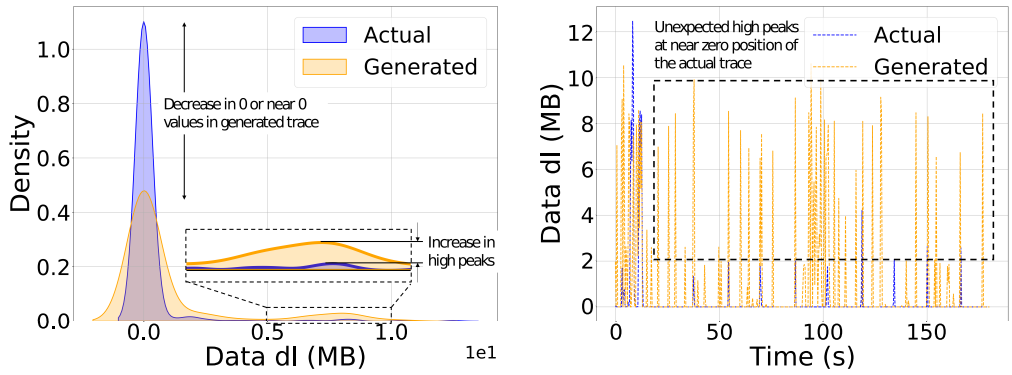
trace—where the majority of the data is downloaded (Fig. 10.8b). Also, Fig. 10.8a indicates that the probabilistic distribution of bytes dl values of traces from VideoTrain are closer to the actual trace distribution than the *GAN-Tunnel* method.

Fig. 10.9 shows the cosine similarity between the actual and corresponding synthesized traces. Cosine similarity has been a widely used metric to evaluate the similarity between two time series distributions (Cassisi, Montalto, Aliotta, Cannata, & Pulvirenti, 2012). Fig. 10.9a and 10.9b indicate that YT 360° videos have 92.5($\pm$4.4)% of cosine similarity, whereas the value for YT Normal video is around 76.2($\pm$7.4)%. Compared to YT, we see high irregularity in FB, yet it achieves

80.5(83.2)% for FB-360° and FB-Normal videos having standard deviation (sd) around ±13(19)% which is caused by anomaly videos (e.g. FB-360°: video ID-19,38 and FB-Normal:video ID-20, 50). We observe that these anomaly videos have completely different streaming patterns from other videos, as shown in Fig. 10.10a and 10.10b. The mixture of high and low cosine similarities in actual FB-Normal videos cause the WGAN model to generate unmatched traces to actual traces.

Therefore, we present an approach to generate time-series data with modifications to the proposed technique in Chapter 7. In this particular work, we apply the proposed VideoTrain framework for video traces generation. The results validate the effectiveness and accuracy of the time-series generation with the capability of synthesizing a statistically similar dataset and outperforms the existing time-series generation apparoaches.

10.5 Trace splitting algorithm

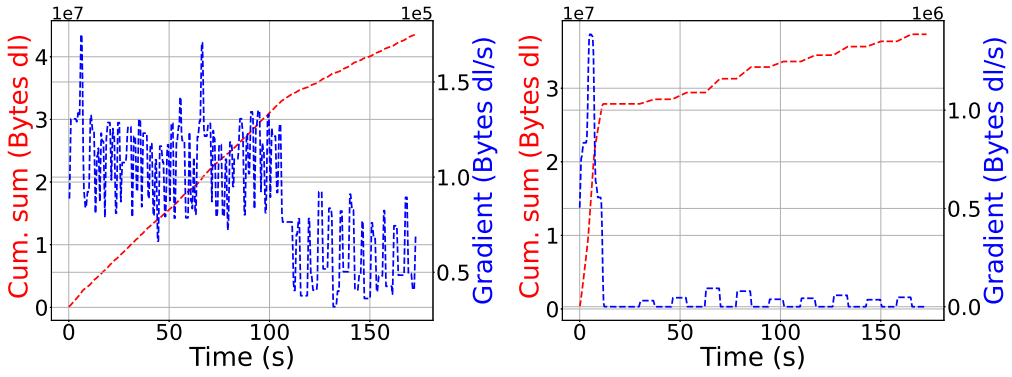


(a) KDE distribution **(b)** Temporal distribution

Figure 10.11: Unexpected peaks between 5–10 MB are observed in KDE distribution (Fig. 10.11a). The majority of these peaks appear after 20 s in temporal distribution (Fig. 10.11b)

We further extend our contribution by introducing a trace splitting algorithm for training data. In our initial experiments, we observed high data throughput with high bursts at the beginning of video streaming session compared to the rest of the session. We experienced that leads to low fidelity in synthesized traces. For example, Fig. 10.11a, which shows the KDE (Kernal Density Estimation) diagram of a sample pair of actual and synthesized trace from **D2**–Netflix, has undesired

peaks around 7.5MB. The main reason is that the actual trace transmits a high amount of data (both in ul and dl) within the first 20–25 Seconds, and a very less amount of data after that. Therefore, the data generation model, which tries to match the high peaks of the signal at the beginning also tends to create unexpected peaks in the remaining part of the trace after 25 Seconds as shown in Fig. 10.11b.



(a) D2–YouTube **(b) D2**–Netflix

Figure 10.12: Moving average of cumulative sum and its gradient distribution for a trace **D2**–YouTube and **D2**–Netflix.

To address this challenge, we propose a mechanism to split the trace based on the cumulative sum of *Tot bytes dl* feature and its gradient. We use these distributions to first calculate a margin in the temporal domain to separate the trace to potential high and low data transmission regions, and then, feed them separately for the data synthesis. Fig. 10.12 shows the cumulative sum and its gradient of a trace shown from Youtube (YT) and Netflix datasets. The cumulative sum distribution of a Netflix trace (Fig. 10.12b) illustrates a significant change in the gradient after the initial data burst and we can clearly decide the margin to split. Contrarily, an Youtube trace (Fig. 10.12a) shows an approximately linear curve throughout the entire video duration making it is difficult to identify a margin to split the trace.

After deciding the margin, we split the trace and train two separate WGAN networks. Then, the synthesized traces are merged back to get the complete synthesized trace. With this approach,

we overcome the challenge of generating video traces with high data transmission at the beginning and generate high-fidelity video traces for classification.

Chapter 11

Summary

11.1 Conclusion

With challenges to identifying latent and emergent behavioral patterns of individuals and groups, homeland security, counterterrorism professionals, and other investigative authorities are actively exploring efficient and scalable computing tools to detect violent extremists in advance. In this work, we presented the INSPECT framework, a set of computation tools integrated to investigative pattern detection that allows flagging suspicious individual and group profiles. We discussed the functionalities and techniques used in each component of INSPECT framework: data extraction in-text sources, data storage, query graph formation, investigative graph search, and synthetic profile generation. We further explained the generalizability and the applicability for diverse datasets and contexts with the modularized architecture.

We integrated different NLP approaches such as named entity recognition (NER), coreference resolution, and multi-label text classification to extract information of behavioral indicators. Additionally, several text pre-processing approaches were used to improve the accuracy of data extraction with smaller and label-wise imbalanced training datasets. Then, we explained the effectiveness of applying a transfer learning-based, pre-trained NLP model called BERT with our small and irregular datasets to achieve significant improvement in information extraction using the multi-label text classification technique. We were able to achieve over 80-85% classification accuracy using multi-label text classification with the BERT model.

We presented the details of how the extracted information from text sources can be modeled to rich knowledge networks. Then, we discussed the importance of utilizing graph databases to store such high-connected data to be stored, updated, and continually examined for the emergence of patterns in the long term. We presented our software routines for investigative graph search that implemented as custom stored procedures on top of graph databases. Additionally, we

demonstrated the wide-range applicability with different datasets and the enhanced effectiveness of observing suspicious or latent trends using our investigative graph searches. We verified that our algorithms were sufficiently capable to provide results within an acceptable time frame against different sizes of datasets. Additionally, the enhancements to the algorithms were presented using the temporal details and multi-occurrences of behavioral indicators for the accurate detection of emergent patterns. Furthermore, we implemented a novel library to convert relational databases to graph databases at scale and efficiency, which facilitates developers and data scientists to transform their datasets to graph databases easily.

In this work, we identified the challenges with small and sparse datasets, which are inherent in behavioral pattern data. A generative adversarial network (GAN) based approach was implemented with novel feature mapping techniques to synthesize data from small and sparse datasets while preserving the statistical characteristics of actual datasets. We expanded our contribution by proposing a novel feature mapping technique for sparse and incomplete datasets to model such data, and apply both likelihood and deep-generative approaches to make it capable of generating high-fidelity data. Gaussian and R-vine copulas were utilized as the likelihood methods and an adversarial autoencoder was integrated as the deep-generative method for the data generation. We also presented a comparative analysis between likelihood and deep-generative approaches using multi-variate data simulation and three different real-world datasets. Then, we applied the proposed data generation technique to synthesize radicalization trajectories using a real-world dataset from WJP (Klausen et al., 2020), which was beneficial for social science studies. The statistical similarity between the generated and actual data was demonstrated via diverse descriptive statistics. We verified that our proposed method outperforms when the dataset is smaller and with more attributes/features (high-dimensional).

Furthermore, we applied our GAN-based data generation approach to two other applications. In the first application, we demonstrated how our data generation approach can be used to synthesize samples that mimic real phishing websites' data. We verified that the classification algorithms included with synthesized data were also significantly more robust to exploratory phishing attacks.

In the second application, our synthetic data generation was applied for a video traffic classification technique. In this work, we adjusted our data generation technique to capture the temporal patterns in time series data. Then, we proposed a WGAN based data generation technique to increase training data as an alternative to internet traffic data collection. We validated that deep learning-based classifiers performed better and enhanced the accuracy by 5-15% with the synthesized data.

With all of these components in INSPECT framework, we streamline investigative pattern detection while addressing scalability and efficiency challenges. We validated each component of INSPECT along with the human-in-the-loop (HITL) process where quantitative and qualitative feedback was provided by social/political scientists iteratively to improve our tools and techniques. We also made a notable contribution to discrete and sparse data generation in various contexts with our novel generative adversarial network-based data synthesizing techniques.

11.2 Future directions

The implemented INSPECT framework further will be applied to other investigative domains while maintaining accuracy and scalability. We also plan to implement a web application to embed the framework with user-friendly interfaces that will significantly benefit social scientists and investigators. We will further enhance the accuracy and efficacy of each component of the framework. The timestamp information extraction that binds to a particular behavioral indicator will be automated. In graph searches, we will develop multi-threaded procedures to improve the query performance on graph databases. One of the proposed multi-threaded approaches is depicted in Fig. 11.1. Additionally, we will study applying semi-supervised methods like *node2vec* to learn feature representations for nodes that obtain more significant insights into the deeper structure of an investigative graph.

The proposed data generation technique will be applied to other domains that require enhancing smaller datasets. We will further evaluate the impact of synthesizing data in different domains and will experiment on how the synthesized data positively or negatively affect particular datasets in general. Further, a quantitative analysis of privacy-preserving through our data generation ap-

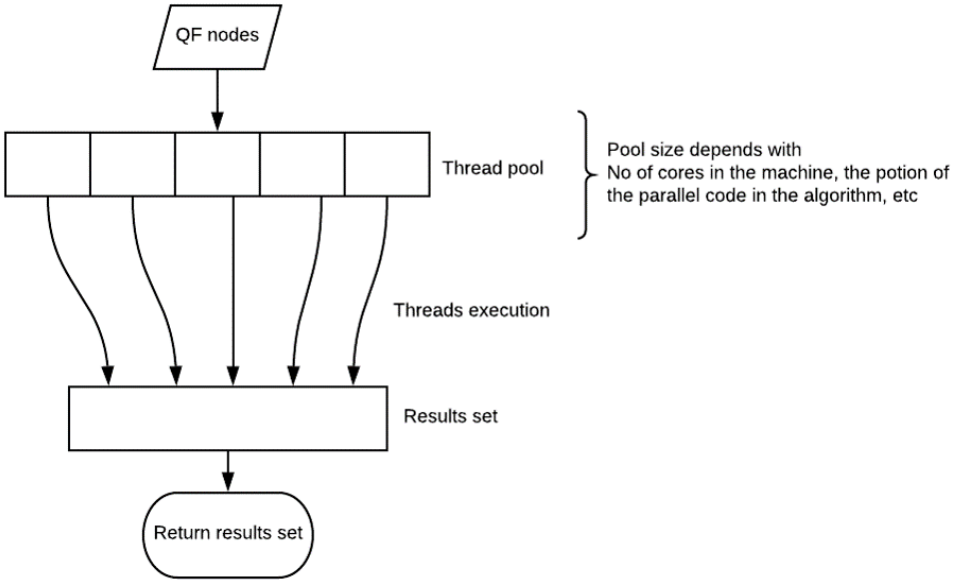


Figure 11.1: Proposed approach for multi-threaded Neo4j stored procedures, Query Focus (QS) node to be run in parallel.

proaches will be investigated. Moreover, we will extend our work to address data anonymization and data privacy with our data generation technique.

Bibliography

Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based associative classification data mining. *Expert Systems with Applications*.

Abt, S., & Baier, H. (2014). Are we missing labels? a study of the availability of ground-truth in network security research. In *2014 third international workshop on building analysis datasets and gathering experience returns for security (badgers)* (pp. 40–55).

Aceto, G., Ciuonzo, D., Montieri, A., & Pescapé, A. (2019). Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management*, *16*(2), 445–458.

Afzal, S., Chen, J., & Ramakrishnan, K. (2017). Characterization of 360-degree videos. In *Proceedings of the workshop on virtual reality and augmented reality network* (pp. 1–6).

Agafonkin, V. (2020). *Leaflet a javascript library for interactive maps.* `https://leafletjs.com`. ([Online]. Accessed on 17-August-2020)

Ahmed, E., Yaqoob, I., Hashem, I. A. T., Shuja, J., Imran, M., Guizani, N., & Bakhsh, S. T. (2018). Recent advances and challenges in mobile big data. *IEEE Communications Magazine*, *56*(2), 102–108.

Amazon Web Services (AWS). (2020). *Database caching strategies using redis.* `https://d0.awsstatic.com/whitepapers/Database/database-caching-strategies-using-redis.pdf`. ([Online]. Accessed on 05-April-2020)

Amazon Web Services (AWS). (2021). *Amazon Comprehend Medical.* https://aws.amazon.com/comprehend/medical/. ([Accessed on 16-Mar-2021])

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Asiler, M., & Yazıcı, A. (2017). Bb-graph: A subgraph isomorphism algorithm for efficiently querying big graph databases. *arXiv preprint arXiv:1706.06654*.

Aydın, B., Pataki, G., Wang, H., Ladha, A., Bullitt, E., & Marron, J. (2012). New approaches to principal component analysis for trees. *Statistics in Biosciences*, *4*(1), 132–156.

Beaulieu-Jones, B. K., Wu, Z. S., Williams, C., Lee, R., Bhavnani, S. P., Byrd, J. B., & Greene, C. S. (2018). Privacy-preserving generative deep neural networks support clinical data sharing. *bioRxiv*. Retrieved from `https://www.biorxiv.org/content/early/2018/12/20/159756` doi: 10.1101/159756

Behavioral Threat Assessment Center of Federal Bureau of Investigation (FBI). (2021). *Lone offender terrorism report.* https://www.fbi.gov/file-repository/lone-offender-terrorism-report-111319.pdf/view. ([Accessed on 01-Apr-2021])

Bellutta, D., Chen, Y., Gartenstein-Ross, D., Pulice, C., Subasic, A., & Subrahmanian, V. S. (2020). Understanding shifting triadic relationships in the al-qaeda/isis faction ecosystem. *IEEE Transactions on Computational Social Systems*, *7*(6), 1423-1434. doi: 10.1109/TCSS.2020.3022586

Blevins, T., Kwiatkowski, R., Macbeth, J., McKeown, K., Patton, D., & Rambow, O. (2016). Automatically processing tweets from gang-involved youth: towards detecting loss and aggression. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 2196–2206).

Bonchi, F., Castillo, C., Gionis, A., & Jaimes, A. (2011). Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 1–37. doi: 10.1145/1961189.1961194

Borum, R. (2012, jun). Radicalization into violent extremism i: A review of social science theories. In *Journal of strategic security* (p. 7-36). doi: 10.5038/1944-0472.4.4.1

Brophy, E., Wang, Z., & Ward, T. E. (2019). Quick and easy time series generation with established image-based gans. *arXiv preprint arXiv:1902.05624*.

Brynielsson, J., Horndahl, A., Johansson, F., Kaati, L., Mårtenson, C., & Svenson, P. (2013). Harvesting and analysis of weak signals for detecting lone wolf terrorists. *Security Informatics*, *2*(1), 1–15.

Campedelli, G. M., Bartulovic, M., & Carley, K. M. (2019). Pairwise similarity of jihadist groups in target and weapon transitions. *Journal of Computational Social Science*, *2*(2), 245–270.

Campedelli, G. M., Cruickshank, I., & Carley, K. M. (2019). A complex networks approach to find latent clusters of terrorist groups. *Applied Network Science*, *4*(1), 59.

Capellan, J., & Lewandowski, C. (2018). Can threat assessment help police prevent mass public shootings? testing an intelligence-led policing tool. *Policing: An International Journal of Police Strategies and Management*, 16–30.

Carletti, V., Foggia, P., Saggese, A., & Vento, M. (2017). Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with vf3. *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 804–818.

Carley, K., Pfeffer, J., Morstatter, F., & Liu, H. (2014, January). Embassies burning: toward a near-real-time assessment of social media using geo-temporal dynamic network analytics. *Social Network Analysis and Mining*, *4*(1), 1–23. doi: 10.1007/s13278-014-0195-3

Cassisi, C., Montalto, P., Aliotta, M., Cannata, A., & Pulvirenti, A. (2012). Similarity measures and dimensionality reduction techniques for time series data mining. *Advances in data mining knowledge discovery and applications*, 71–96.

Castellani, B., Griffiths, F., Rajaram, R., & Gunn, J. (2018). Exploring comorbid depression and physical health trajectories: A case-based computational modelling approach. *Journal of evaluation in clinical practice*, *24*(6), 1293–1309.

CERT Division Software Engineering Institute. (2019). *Insider threat best practices.* `https://resources.sei.cmu.edu/library/asset-view.cfm`. ([Online]. Accessed on 28-June-2019)

Chen, G., & Wang, G. (2020). A supervised learning algorithm for spiking neurons using spike train kernel based on a unit of pair-spike. *IEEE Access*.

Cherubini, U., Gobbi, F., & Mulinacci, S. (2016). *Convolution copula econometrics*. Springer.

Choi, E., Biswal, S., Malin, B., Duke, J., Stewart, W. F., & Sun, J. (2017). Generating multi-label discrete patient records using generative adversarial networks. *arXiv preprint*

arXiv:1703.06490.

Choi, K., Wijesinghe, A., Kattadige, C., Thilakarathna, K., Seneviratne, S., & Jourjon, G. (2020). Seta: Scalable encrypted traffic analytics inmulti-gbps networks. In *2020 ieee 17th international conference on local computer networks (lcn)*.

Cisco. (2019, February). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022* (Tech. Rep.). CA, USA: Author.

Clark, A., Donahue, J., & Simonyan, K. (2019). Adversarial video generation on complex datasets. *arXiv*, arXiv–1907.

Computer Network Research Laboratory (CNRL), Colorado State University. (2022a). *Bert multi-label text classification by PyTorch (Customized for WJP radicalization datasets).* `https://github.com/cnrl-csu/rad-bert`. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022b). *Complex data object generation with Adversarial Autoencoders (AAE) and Copulas.* `https://github.com/cnrl-csu/complex-object-datagen`. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022c). *Multi-label Text Classification Analysis.* `https://github.com/cnrl-csu/multi-label-classification`. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022d). *Network data generation via Wasserstein GANs and percentile distribution mapping.* `https://github.com/cnrl-csu/network_datagen`. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022e). *Phishing websites data generation using Adversarial Autoencoders (AAE).* `https://github.com/cnrl-csu/phishing-datagen`. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022f). *PINGS 1.0 (Procedures for INvestigative Graph Search).* `https://github.com/cnrl-csu/`

pings. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022g). *PINGS (Procedures for INestigative Graph Search) Library.* http://www.cnrl.colostate.edu/Projects/RAD/pings.html. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022h). *Radicalization trajectory generation using Adversarial Autoencoders (AAE) and Copulas.* https://github.com/cnrl-csu/rad-datagen. ([Online]. Accessed on 03-Feb-2022)

Computer Network Research Laboratory (CNRL), Colorado State University. (2022i). *Relational to neo4j graph database conversion (rel2neo).* https://github.com/cnrl-csu/rel2neo. ([Online]. Accessed on on 31-Jan-2022)

Cordella, L. P., Foggia, P., Sansone, C., & Vento, M. (2004). A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, *26*(10), 1367–1372.

Czado, C. (2011). The world of vines. *Technische Universität München*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, jun). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. doi: 10.18653/v1/N19-1423

Dhamija, R., Tygar, J. D., & Hearst, M. (2006). Why phishing works. In *Conference on human factors in computing systems.*

Dheeru, D., & Karra Taniskidou, E. (2019). *UCI machine learning repository.* (http://archive.ics.uci.edu/ml (Accessed 2019-05-12))

Dimopoulos, G., Leontiadis, I., Barlet-Ros, P., & Papagiannaki, K. (2016). Measuring video qoe from encrypted traffic. In *Proceedings of the 2016 internet measurement conference* (pp.

513–526).

DOJ Public Release. (2021). *Individual who planned attack in queens charged with attempting to provide material support to isis.* https://www.justice.gov/opa/pr/individual-who-planned-attack-queens-charged-attempting-provide-material-support-isis. ([Online]. Accessed on 31-Aug-2021)

Donahue, C., McAuley, J., & Puckette, M. (2018). Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*.

Dou, Z., Khalil, I., Khreishah, A., Al-Fuqaha, A., & Guizani, M. (2017). Systematization of knowledge (sok): A systematic review of software-based web phishing detection. *IEEE Communications Surveys Tutorials*.

Edelman, D., & Singer, M. (2015, November). Competing on customer journeys. *Harvard Business Review*.

Edelman, D., & Singer, M. (2019). *The new consumer decision journey.* `https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-new-consumer-decision-journey`. ([Online]. Accessed on 28-June-2019)

Facebook. (2020). *Facebook graph search.* `https://www.facebook.com/graphsearcher/`. ([Online]. Accessed on 09-March-2020)

Fan, W., Wang, X., & Wu, Y. (2013). Diversified top-k graph pattern matching. *Proceedings of the VLDB Endowment*, *6*(13), 1510–1521.

Fathi-Kazerooni, S., & Rojas-Cessa, R. (2020). Gan tunnel: Network traffic steganography by using gans to counter internet traffic classifiers. *IEEE Access*, *8*, 125345–125359.

Federal Bureau of Investigation (FBI). (2020a). *Business e-mail compromise 12 billion dollar scam.* (`https://www.ic3.gov/media/2018/180712.aspx`, (accessed July 2, 2020))

Federal Bureau of Investigation (FBI). (2020b). *Gangs.* `https://www.fbi.gov/investigate/violent-crime/gangs`. ([Online]. Accessed on 11-April-2020)

Figueroa, R. L., Zeng-Treitler, Q., Kandula, S., & Ngo, L. H. (2012). Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*.

Folch-Fortuny, A., Villaverde, A. F., Ferrer, A., & Banga, J. R. (2015). Enabling network inference methods to handle missing data and outliers. *BMC Bioinformatics*, *16*(1), 1–12. Retrieved from http://dx.doi.org/10.1186/s12859-015-0717-7 doi: 10.1186/s12859-015-0717-7

Freeland, J., Klausen, J., & Pagé, C. A. (2019). *Dynamic threat assessment: An innovative approach to preventive investigations of violent extremism and terrorism* (Martine Evans and Massil Benbouriche ed.). Lexington Books.

Freilich, J. D., Chermak, S. M., Gruenewald, R. B. J., & Parkin, W. S. (2014). Introducing the united states extremist crime database (ecdb). *Terrorism and Political Violence*, *26*(2), 372-384. Retrieved from https://doi.org/10.1080/09546553.2012.713229 doi: 10.1080/09546553.2012.713229

Friedman, C., Rindflesch, T. C., & Corn, M. (2013). Natural language processing: state of the art and prospects for significant progress, a workshop sponsored by the national library of medicine. *Journal of biomedical informatics*, *46*(5), 765–773.

Ganesan, M., Prashant, S., & Jhunjhunwala, A. (2012). A review on challenges in implementing mobile phone based data collection in developing countries. *Journal of Health Informatics in Developing Countries*, *6*(1).

Genest, C., Gendron, M., & Bourdeau-Brien, M. (2009). The advent of copulas in finance. *The European Journal of Finance*, *15*(7-8), 609–618.

A gentle introduction to transfer learning for deep learning. (n.d.). https://machinelearningmastery.com/transfer-learning-for-deep-learning/. ([Online]. Accessed on 12-May-2020)

Gianfrancesco, M. A., Tamang, S., Yazdany, J., & Schmajuk, G. (2018). Potential biases in machine learning algorithms using electronic health record data. *JAMA internal medicine*, *178*(11), 1544–1547.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014, Dec). Generative adversarial networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*, *2*, 2672-2680.

Government Digital Service UK. (2020). *UK open data.* `https://data.gov.uk/`. ([Online]. Accessed on 09-March-2020)

Guan, Y., Zheng, C., Zhang, X., Guo, Z., & Jiang, J. (2019). Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the acm special interest group on data communication* (pp. 394–407).

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, *30*, 5767–5777.

Gutterman, C., Guo, K., Arora, S., Wang, X., Wu, L., Katz-Bassett, E., & Zussman, G. (2019). Requet: Real-time qoe detection for encrypted youtube traffic. In *Proc. of the 10th acm mmsys* (pp. 48–59).

Handa, A., Sharma, A., & Shukla, S. K. (2019). Machine learning in cybersecurity: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(4), e1306.

Hlaoui, A., & Wang, S. (2002). A new algorithm for inexact graph matching. In *Object recognition supported by user interaction for service robots* (Vol. 4, pp. 180–183).

Ho, G., Cidon, A., Gavish, L., Schweighauser, M., Paxson, V., Savage, S., ... Wagner, D. (2019). Detecting and characterizing lateral phishing at scale. In *Usenix security symposium.*

Hoffman, B. (n.d.). *'isis' shifting focus.* (`https://www.thecipherbrief.com/column_article/isis-shifting-focus` (Accessed 2019-04-30))

Hoffman, B., Meese, E., & Roemer, T. (2015). *The FBI: Protecting the homeland in the 21st century* (Tech. Rep.). USA: Federal Bureau of Investigation (FBI). https://www.hsdl.org/?abstract&did=763412. ([Accessed on 1-April-2020])

Holford, M. (2022). *Let's write a stored procedure in neo4j – part i.* https://neo4j.com/developer-blog/lets-write-a-stored-procedure-in-neo4j-part-i/. ([Online]. Accessed on 03-Feb-2022)

Homeland Security Committee. (2018). Terror gone viral- overview of the 243 isis-linked incidents targeting the west..

Hong, J., Kim, T., Liu, J., Park, N., & Kim, S.-W. (2020). Phishing url detection with lexical features and blacklisted domains. In *Adaptive autonomous secure cyber systems* (pp. 253–267). Springer.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., & Tygar, J. D. (2011). Adversarial machine learning. In *Acm workshop on security and artificial intelligence.*

Hung, B., Jayasumana, A., & Bandara, V. (2017). Insight: Detecting the radicalization trajectories of homegrown violent extremists with dynamic graph pattern matching. In *Ieee homeland security technologies (hst) symposium 2017 conference proceedings.*

Hung, B. W., & Jayasumana, A. P. (2016). Investigative simulation: Towards utilizing graph pattern matching for investigative search. In *2016 ieee/acm international conference on advances in social networks analysis and mining (asonam)* (pp. 825–832).

Hung, B. W., Jayasumana, A. P., & Bandara, V. W. (2018). INSiGHT: A system to detect violent extremist radicalization trajectories in dynamic graphs. *Data & Knowledge Engineering*, *118*, 52–70.

Hung, B. W., Jayasumana, A. P., & Bandara, V. W. (2019). Finding emergent patterns of behaviors in dynamic heterogeneous social networks. *IEEE Transactions on Computational Social Systems*, *6*(5), 1007–1019.

Hung, B. W., Muramudalige, S. R., Jayasumana, A. P., Klausen, J., Libretti, R., Moloney, E., & Renugopalakrishnan, P. (2019, Nov). Recognizing radicalization indicators in text documents using human-in-the-loop information extraction and nlp techniques. In *2019 ieee international symposium on technologies for homeland security (hst)* (pp. 1–7).

Hung, B. W. K., Muramudalige, S. R., Jayasumana, A. P., Klausen, J., Libretti, R., Moloney, E., & Renugopalakrishnan, P. (2019). Recognizing radicalization indicators in text documents using human-in-the-loop information extraction and nlp techniques. In *2019 ieee international symposium on technologies for homeland security (hst)* (p. 1-7).

Islam, K. T., Shelton, C. R., Casse, J. I., & Wetzel, R. (2017). Marked point process for severity of illness assessment. In *Machine learning for healthcare conference* (pp. 255–270).

Jain, A. K., & Gupta, B. B. (2018). Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*.

Jammazi, R., & Reboredo, J. C. (2016). Dependence and risk management in oil and stock markets. a wavelet-copula analysis. *Energy*, *107*, 866–888.

Janson, S. (2018). On edge exchangeable random graphs. *Journal of statistical physics*, *173*(3), 448–484.

Jashinsky, J., Burton, S. H., Hanson, C. L., West, J., Giraud-Carrier, C., Barnes, M. D., & Argyle, T. (2014). Tracking suicide risk factors through twitter in the us. *Crisis*, *35*, 51–59.

Jaworski, P., Durante, F., Hardle, W. K., & Rychlik, T. (2010). *Copula theory and its applications* (Vol. 198). Springer.

Jiang, S., Ma, Z., Zeng, X., Xu, C., Zhang, M., Zhang, C., & Liu, Y. (2020). Scylla: QoE-aware continuous mobile vision with fpga-based dynamic deep neural network reconfiguration. In *Ieee infocom* (pp. 1369–1378).

Joe, H., Li, H., & Nikoloulopoulos, A. K. (2010). Tail dependence functions and vine copulas. *Journal of Multivariate Analysis*, *101*(1), 252 - 270. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0047259X09001481` doi: https://doi.org/10.1016/j.jmva.2009.08.002

Johnson, A. E., Pollard, T. J., Shen, L., Li-Wei, H. L., Feng, M., Ghassemi, M., ... Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, *3*(1), 1–9.

Joshi, M., Levy, O., Weld, D. S., & Zettlemoyer, L. (2019). Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*.

Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vanhaelen, Q., Khrabrov, K., & Zhavoronkov, A. (2017). The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, *8*(7), 10883.

Kanezashi, H., Suzumura, T., Garcia-Gasulla, D., Oh, M. H., & Matsuoka, S. (2019). Adaptive Pattern Matching with Reinforcement Learning for Dynamic Graphs. In *Proceedings - 25th ieee international conference on high performance computing, hipc 2018* (pp. 92–101). doi: 10.1109/HiPC.2018.00019

Kapur, R. (2019). *Sri lanka's easter sunday bombings: Moving forward.* https://www.mei.edu/publications/sri-lankas-easter-sunday-bombings-moving-forward. ([Accessed on 30-Oct-2019])

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4401–4410).

Keras applications. (n.d.). https://keras.io/api/applications/. ([Online]. Accessed on 12-May-2021)

Khan, A., Wu, Y., Aggarwal, C. C., & Yan, X. (2013). NeMa: Fast graph search with label similarity. *Proceedings of the VLDB Endowment*, *6*(3), 181–192. doi: 10.14778/2535569.2448952

King, M., & Taylor, D. M. (2011). The radicalization of homegrown jihadists: A review of theoretical models and social psychological evidence. *Terrorism and political violence*, *23*(4), 602–622.

Kirchner, J., Heberle, A., & Löwe, W. (2015). Classification vs. regression-machine learning approaches for service recommendation based on measured consumer experiences. In *Ieee world congress on services.*

Klausen, J. (2016). *A behavioral study of the radicalization trajectories of american" homegrown" al qaeda-inspired terrorist offenders*. Brandeis University.

Klausen, J., Campion, S., Needle, N., Nguyen, G., & Libretti, R. (2016). Toward a behavioral model of "homegrown" radicalization trajectories. *Studies in Conflict & Terrorism*, *39*(1), 67-83. doi: 10.1080/1057610X.2015.1099995

Klausen, J., Libretti, R., Hung, B. W. K., & Jayasumana, A. P. (2018). Radicalization trajectories: An evidence-based computational approach to dynamic risk assessment of "homegrown" jihadists. *Studies in Conflict & Terrorism*, 1-28. doi: 10.1080/1057610X.2018.1492819

Klausen, J., Libretti, R., Renugopalakrishnan, P., & et al. (2020). *Jytte klausen's western jihadism project.* `https://www.brandeis.edu/klausen-jihadism/`. ([Online] Accessed on 26-Jan-2020)

Klausen, J., Marks, C. E., & Zaman, T. (2018, August). Finding extremists in online social networks. *Oper. Res.*, *66*(4), 957–976. Retrieved from `https://doi.org/10.1287/opre.2018.1719` doi: 10.1287/opre.2018.1719

Kreimeyer, K., Foster, M., Pandey, A., Arya, N., Halford, G., Jones, S. F., ... Botsis, T. (2017). Natural language processing systems for capturing and standardizing unstructured clinical information: a systematic review. *Journal of biomedical informatics*, *73*, 14–29.

Lara-Cabrera, R., Pardo, A. G., Benouaret, K., Faci, N., Benslimane, D., & Camacho, D. (2017). Measuring the radicalisation risk in social networks. *IEEE Access*, *5*, 10892–10900.

Lawrence, S., & Giles, C. L. (2000). Overfitting and neural networks: conjugate gradient and backpropagation. In *Proc. of the ijcnn 2000* (Vol. 1, pp. 114–119).

Lee, I., & Shin, Y. J. (2020). Machine learning for enterprises: Applications, algorithm selection, and challenges. *Business Horizons*, *63*(2), 157–170.

Li, F., Chung, J. W., & Claypool, M. (2018). Silhouette: Identifying youtube video flows from encrypted traffic. In *Proceedings of the 28th acm sigmm workshop on network and operating systems support for digital audio and video* (pp. 19–24).

Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Li, Y., Huang, Y., Xu, R., Seneviratne, S., Thilakarathna, K., Cheng, A., ... Jourjon, G. (2018). Deep content: Unveiling video streaming content from encrypted wifi traffic. In *2018 ieee 17th international symposium on network computing and applications (nca)* (pp. 1–8).

Lin, Z., Jain, A., Wang, C., Fanti, G., & Sekar, V. (2019). Generating high-fidelity, synthetic time series datasets with doppelganger. *arXiv preprint arXiv:1909.13403*.

Lindekilde, L., O'Connor, F., & Schuurman, B. (2019). Radicalization patterns and modes of attack planning and preparation among lone-actor terrorists: an exploratory analysis. *Behavioral Sciences of Terrorism and Political Aggression*, *11*(2), 113–133.

Liu, L., Zhong, R., Zhang, W., Liu, Y., Zhang, J., Zhang, L., & Gruteser, M. (2018). Cutting the cord: Designing a high-quality untethered vr system with low latency remote rendering. In *Proc. of the 16th mobisys* (pp. 68–80).

Ly, S., Pho, K.-H., Ly, S., & Wong, W.-K. (2019). Determining distribution for the product of random variables by using copulas. *Risks*, *7*(1), 23.

Ma, S., Cao, Y., Fan, W., Huai, J., & Wo, T. (2014). Strong simulation: Capturing topology in graph pattern matching. *ACM Transactions on Database Systems (TODS)*, *39*(1), 1–46.

Machine-learning Mastery. (2021). *Sensitivity analysis for k.* https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/. ([Online]. Accessed on 23-Sep-2021)

MacNeil Vroomen, J., Eekhout, I., Dijkgraaf, M. G., van Hout, H., de Rooij, S. E., Heymans, M. W., & Bosmans, J. E. (2016). Multiple imputation strategies for zero-inflated cost data in economic evaluations: which method works best? *The European Journal of Health Economics*, *17*(8), 939–950. Retrieved from `https://doi.org/10.1007/s10198-015-0734-5` doi: 10.1007/s10198-015-0734-5

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Mancini, L., & Paganoni, A. M. (2019). Marked point process models for the admissions of heart failure patients. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *12*(2), 125–135.

Mao, J., Bian, J., Tian, W., Zhu, S., Wei, T., Li, A., & Liang, Z. (2019). Phishing page detection via learning classifiers from page layout feature. *EURASIP Journal on Wireless Communi-*

cations and Networking.

MITRE. (2021). *Person-centric identity management-assimilating data.* https://www.mitre.org/publications/technical-papers/person-centric-identity-management-rapidly-assimilating-data-about-a. ([Online]. Accessed on 31-May-2021)

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2012). An assessment of features related to phishing websites using an automated technique. In *Internet technology and secured transactions.*

Muramudalige, S. R., Hung, B. W., Jayasumana, A. P., & Ray, I. (2019). Investigative graph search using graph databases. In *2019 first international conference on graph computing (gc)* (pp. 60–67).

Muramudalige, S. R., Hung, B. W. K., Jayasumana, A. P., Ray, I., & Klausen, J. (2021). Enhancing investigative pattern detection via inexact matching and graph databases. *IEEE Transactions on Services Computing*. Retrieved from `https://doi.org/10.1109/TSC.2021.3073145` doi: 10.1109/TSC.2021.3073145

National Institutes of Health, U. S. D. o. H., & Services, H. (2020). *Health insurance portability and accountability act (hipaa) privacy rule.* `https://privacyruleandresearch.nih.gov/`. ([Online] Accessed on 20-May-2020)

Needham, M., & Hodler, A. (2019). *Graph algorithms*. O'Reilly Media, Inc.

Neo4j.com. (2019a). *The neo4j graph algorithms user guide v3.5.* `https://neo4j.com/docs/graph-algorithms/current/`. ([Online]. Accessed on 20-July-2019)

Neo4j.com. (2019b). *Neo4j graph platform.* `https://neo4j.com/`. ([Online]. Accessed on 17-June-2019)

Neo4j.com. (2020a). *Neo4j graph database sandbox.* `https://neo4j.com/sandbox-v2/`. ([Online]. Accessed on 09-March-2020)

Neo4j.com. (2020b). *Neo4j performance tuning.* `https://neo4j.com/developer/guide-performance-tuning/`. ([Online]. Accessed on 05-April-2020)

Neo4j.com. (2020c). *What is a graph database?* https://neo4j.com/developer/graph-database/. ([Online]. Accessed on 09-March-2020)

Neo4j.com. (2022). *User-defined functions.* https://neo4j.com/docs/developer-manual/current/extending-neo4j/cypher-functions/. ([Online]. Accessed on 03-Feb-2022)

Nesser, P., Stenersen, A., & Oftedal, E. (2017). *Jihadi terrorism in europe: The is-effect.* http://www.terrorismanalysts.com/pt/index.php/pot/article/view/553/html. ([Accessed on 30-Oct-2019])

Niakanlahiji, A., Chu, B.-T., & Al-Shaer, E. (2018). Phishmon: A machine learning framework for detecting phishing webpages. In *Intelligence and security informatics.*

Olmos, I., Gonzalez, J. A., & Osorio, M. (2006). Inexact graph matching: A case of study. In *Flairs conference* (pp. 586–591).

Olson, R. (2011). *Suicide threats on social network sites centre for suicide prevention.* http://www.sprc.org/resources-programs/suicide-threats-social-networking-sites. ([Online]. Accessed on 28-June-2019)

Panamtash, H., Zhou, Q., Hong, T., Qu, Z., & Davis, K. O. (2020). A copula-based bayesian method for probabilistic solar power forecasting. *Solar Energy*, *196*, 336–345.

Parisi, F., Park, N., Pugliese, A., & Subrahmanian, V. S. (2018). Top-k user-defined vertex scoring queries in edge-labeled graph databases. *ACM Transactions on the Web*, *12*(4). doi: 10.1145/3213891

Patil, V., Thakkar, P., Shah, C., Bhat, T., & Godse, S. (2018). Detection and prevention of phishing websites using machine learning approach. In *International conference on computing communication control and automation.*

Peng, C., Xu, M., Xu, S., & Hu, T. (2017). Modeling and predicting extreme cyber attack rates via marked point processes. *Journal of Applied Statistics*, *44*(14), 2534–2563.

R. E. Travers, National Counterterrorism Center. (2019). *Counterterrorism in an era of competing priorities.* https://www.dni.gov/files/NCTC/documents/news_documents/Travers_Washington_Institute_Remarks_as

_Prepared.pdf. ([Online]. Accessed on 01-Apr-2021)

Rezaei, S., & Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine*, *57*(5), 76–81.

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from urls. *Expert Systems with Applications*.

Schmitt, P., Bronzino, F., Ayoubi, S., Martins, G., Teixeira, R., & Feamster, N. (2019). Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. *arXiv preprint arXiv:1901.05800*.

Sengar, H., Wang, H., Wijesekera, D., & Jajodia, S. (2008). Detecting voip floods using the hellinger distance. *IEEE transactions on parallel and distributed systems*, *19*(6), 794–805.

Settles, B. (2009). *Active learning literature survey* (Tech. Rep.). WI, USA: University of Wisconsin-Madison Department of Computer Sciences.

Shang, H., Zhang, Y., Lin, X., & Yu, J. X. (2008). Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment*, *1*(1), 364–375.

Sherry, J., Gao, P. X., Basu, S., Panda, A., Krishnamurthy, A., Maciocco, C., ... others (2015). Rollback-recovery for middleboxes. In *Proceedings of the 2015 acm conference on special interest group on data communication* (pp. 227–240).

Shirazi, H., Bezawada, B., & Ray, I. (2018). "kn0w thy doma1n name": Unbiased phishing detection using domain name based features. In *Access control models and technologies.*

Shirazi, H., Bezawada, B., Ray, I., & Anderson, C. (2019). Adversarial sampling attacks against phishing detection. In *Ifip annual conference on data and applications security and privacy.*

Shirazi, H., Muramudalige, S. R., Ray, I., & Jayasumana, A. P. (2020). Improved phishing detection algorithms using adversarial autoencoder synthesized data. In *2020 ieee 45th conference on local computer networks (lcn)* (pp. 438–446).

Sienkiewicz, E., Wang, H., et al. (2018). Pareto quantiles of unlabeled tree objects. *The Annals of Statistics*, *46*(4), 1513–1540.

Silver, J., Simons, A., & Craun, S. (2018, June). *A study of the pre-attack behaviors of active shooters in the united states between 2000 and 2013.* https://www.fbi.gov/file-repository/pre-attack-behaviors-of-active-shooters-in-us-2000-2013.pdf/view. ([Online]. Accessed on 24-May-2021)

Singh, S., Sarje, A. K., & Misra, M. (2012). Client-side counter phishing application using adaptive neuro-fuzzy inference system. In *International conference on computational intelligence and communication networks.*

Smith, A. G. (2018). *How radicalization to terrorism occurs in the united states: What research sponsored by the national institute of justice tells us*. US Department Of Justice, Office of Justice Programs, National Institute of .... https://www.ncjrs.gov/pdffiles1/nij/250171.pdf.

Sukhmani, S., Sadeghi, M., Erol-Kantarci, M., & El Saddik, A. (2018). Edge caching and computing in 5g for mobile ar/vr and tactile internet. *IEEE MultiMedia*, *26*(1), 21–30.

Sun, Y., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Learning vine copula models for synthetic data generation. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 5049–5057).

Tan, C. L. (2018). *Phishing dataset for machine learning: Feature evaluation.* (https://data.mendeley.com/datasets/h3cgnj8hft/1 (Accessed 2019-05-12))

The Associated Press. (2021). *Ireland frees american arrested in terrorist inquiry.* https://www.nwaonline.com/news/2010/mar/14/ireland-frees-american-arrested-terrorist-20100314/. ([Online]. Accessed on 31-May-2021)

The office of director of national intelligence. (2019). *Homegrown violent extremist mobilization indicators.* https://www.dni.gov/files/NCTC/documents/news_documents/NCTC-FBI-DHS-HVE-Mobilization-Indicators-Booklet-2019.pdf.

Times, L. A. (2015, December). *Everything we know about the san bernardino terror attack investigation so far.* https://www.latimes.com/local/california/la-me-san-bernardino-shooting-terror-investigation-htmlstory.html. ([Online]. Accessed on 20-April-2020)

Tong, H., Faloutsos, C., Gallagher, B., & Eliassi-Rad, T. (2007). Fast best-effort pattern matching in large attributed graphs. In *Proceedings of the acm sigkdd international conference on knowledge discovery and data mining* (pp. 737–746). doi: 10.1145/1281192.1281271

Torre, E., Marelli, S., Embrechts, P., & Sudret, B. (2019). A general framework for data-driven uncertainty quantification under complex input dependencies using vine copulas. *Probabilistic Engineering Mechanics*, *55*, 1–16.

Twitter. (2020). *Search tweets.* `https://developer.twitter.com/en/docs/tweets/search/overview`. ([Online]. Accessed on 09-March-2020)

Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, *23*(1), 31–42.

Upadhyay, U., De, A., & Rodriguez, M. G. (2018). Deep reinforcement learning of marked temporal point processes. In *Advances in neural information processing systems* (pp. 3168–3178).

Valverde, M. (2017). *A look at the data on domestic terrorism and who's behind it.* https://www.politifact.com/truth-o-meter/article/2017/aug/16/look-data-domestic-terrorism-and-whos-behind-it/. ([Accessed on 30-Oct-2019])

Vassøy, B., Ruocco, M., de Souza da Silva, E., & Aune, E. (2019). Time is of the essence: a joint hierarchical rnn and point process model for time and item predictions. In *Proceedings of the twelfth acm international conference on web search and data mining* (pp. 591–599).

Wang, H., Marron, J., et al. (2007). Object oriented data analysis: Sets of trees. *The Annals of Statistics*, *35*(5), 1849–1873.

Wang, Z., Qian, Z., Xu, Q., Mao, Z., & Zhang, M. (2011). An untold story of middleboxes in cellular networks. *ACM SIGCOMM Computer Communication Review*, *41*(4), 374–385.

Wang, Z., Wang, W., Liu, C., Wang, Z., & Hou, Y. (2017). Probabilistic forecast for multiple wind farms based on regular vine copulas. *IEEE Transactions on Power Systems*, *33*(1), 578–589.

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? In *Digital image computing: Techniques and*

applications.

Wu, W., Wang, K., Han, B., Li, G., Jiang, X., & Crow, M. L. (2015). A versatile probability model of photovoltaic generation using pair copula construction. *IEEE Transactions on Sustainable Energy*, *6*(4), 1337–1345.

Xiao, S., Farajtabar, M., Ye, X., Yan, J., Song, L., & Zha, H. (2017). Wasserstein learning of deep generative point process models. In *Advances in neural information processing systems* (pp. 3247–3257).

Xiao, S., Xu, H., Yan, J., Farajtabar, M., Yang, X., Song, L., & Zha, H. (2018). Learning conditional generative models for temporal point processes. In *Thirty-second aaai conference on artificial intelligence.*

Xu, J.-M., Jun, K.-S., Zhu, X., & Bellmore, A. (2012). Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 656–666).

Yan, J. (2019). Recent advance in temporal point process: from machine learning perspective. *SJTU Technical Report*.

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first aaai conference on artificial intelligence.*

Zhang, N., Yang, P., Ren, J., Chen, D., Yu, L., & Shen, X. (2018). Synergy of big data and 5g wireless networks: opportunities, approaches, and challenges. *IEEE Wireless Comm.*, *25*(1), 12–18.

Zhang, S., Li, S., & Yang, J. (2009). Gaddi: distance index based subgraph matching in biological networks. In *Proceedings of the 12th international conference on extending database technology: Advances in database technology* (pp. 192–203).

Zhang, X., & Jiang, H. (2019). Application of copula function in financial risk analysis. *Computers & Electrical Engineering*, *77*, 376–388.

Zhang, X., Tong, J., Vishwamitra, N., Whittaker, E., Mazer, J. P., Kowalski, R., ... Dillon, E. (2016). Cyberbullying detection with a pronunciation based convolutional neural network.

In *2016 15th ieee international conference on machine learning and applications (icmla)* (pp. 740–745).

Zhao, T., Wang, J., & Zhang, Y. (2019). Day-ahead hierarchical probabilistic load forecasting with linear quantile regression and empirical copulas. *IEEE Access*, *7*, 80969–80979.

Zhou, D., Yan, Z., Fu, Y., & Yao, Z. (2018). A survey on network data collection. *Journal of Network and Computer Applications*, *116*, 9–23.

Zhou, X., & Verma, R. (2020). Phishing sites detection from a web developer's perspective using machine learning. In *Proceedings of the 53rd hawaii international conference on system sciences.*

Appendix A

Codebase

We implemented different algorithms, tools, and techniques throughout this dissertation to achieve our goals and objectives in Section 3.1. We further describe the existing libraries that we utilized and customizations made in this work.

A.1 Behavioural Indicator Extraction using NLP techniques

For behavioral indication extraction from disperate text sources, we utilized different NLP techniques. We further used text pre-processing techniques to improve the model detection accuracy with our limited datasets.

For Named entity recognition (NER), we applied the Spacy NER, which is a python-based pre-trained NER model[16] to extract different entities from text sources. The further details can be found in Section 5.2. For Coreference resolution, we utilized NeuralCoref[17] library, which was built on top of the Spacy NLP pipeline. More description is available in Section 5.3.

For multi-label text classification that discussed in Section 5.5, we utilized both SpaCy [18] and BERT (Devlin et al., 2018) transfer learning model. In our experiments, we used Pytorch-based BERT transformer [19] and customized for our datasets. The codebase can be found in GitHub repositories (Computer Network Research Laboratory (CNRL), Colorado State University, 2022c, 2022a) with all the steps.

[16]https://spacy.io/api/entityrecognizer

[17]https://github.com/huggingface/neuralcoref

[18]https://spacy.io/api/textcategorizer

[19]https://github.com/lonePatient/Bert-Multi-Label-Text-Classification

A.2 Investigative graph search

We implemented our own algorithms on top of the Neo4j graph database as custom Neo4j procedures (Neo4j.com, 2022; Holford, 2022) to detect suspicious individuals and groups and described in Chapter 6. Currently, Neo4j only supports Java programming language to implement custom procedures and our implemented algorithms were shipped as a JAR (Java ARchive) file [20] and added to the database file system to run our procedures via Neo4j desktop [21] which is an interactive user interface to interact with Neo4j databases.

A.2.1 PINGS

We described the details of our algorithms in Chapter 6 and the code base is publicly available under GNU GPLv3 licence in a GitHub code repository (Computer Network Research Laboratory (CNRL), Colorado State University, 2022f) of Computer Networking Research Laboratory (CNRL), Colorado State University (CSU).

A.3 Rel2Neo - Relational to graph database conversion library

With the identification of lack of tools to convert relational databases to graph databases, we implemented a Python-based tool to convert relational data to Neo4j graph database by facilitating the full control of the conversion to the user. The user can define the required conversion such as nodes, relationships, attributes of nodes and relationships, the direction of a relationship, etc. All details were discussed in Section 6.9 and the codebase is available with examples in CNRL GitHub page (Computer Network Research Laboratory (CNRL), Colorado State University, 2022i).

A.4 Synthetic data generation

We implemented different synthetic data generation schemes using generative adversarial networks (GAN) to various applications that suffered from a lack of data due to specific reasons.

[20] https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jarGuide.html

[21] https://neo4j.com/developer/neo4j-desktop/

The requirements of having such synthetic data generation techniques and the details of different generative approaches were discussed in Chapters 7, 8, 9, and 10. The implemented codebase for synthetic data generation schemes is based on the Tensorflow [22] machine learning library. We implemented a complex object generation technique with the comparison of adversarial auto encoders (AAE) and copula-based synthetic data generation. This approach is applicable in many domains where data is sparse and discrete with our novel feature mapping technique and the details were presented in Chapter 7. The codebase can be found in a repository (Computer Network Research Laboratory (CNRL), Colorado State University, 2022b) of CNRL GitHub page.

We applied AAE-based synthetic data generation mechanism for radicalization trajectory generation and presented in Chapter 8. The codebase is available in a GitHub repository (Computer Network Research Laboratory (CNRL), Colorado State University, 2022h). We also implemented an AAE-based synthetic data generation method to improve phishing websites classification and the details were presented in Chapter 9. The codebase can be found in a GitHub repository (Computer Network Research Laboratory (CNRL), Colorado State University, 2022e). We also implemented Wasserstein-GAN (WGAN) based time-series generation mechanism for video traces to improve the classification accuracy. All the related content was presented in Chapter 10 and the code is available in a CNRL Github repo (Computer Network Research Laboratory (CNRL), Colorado State University, 2022d).

[22] https://www.tensorflow.org