A study on proactive data-driven cyber defense through threat intelligence

アリエル, ロドリゲズ

Kyushu University

A study on proactive data-driven cyber defense through threat intelligence

Ariel Rodriguez

January 2021

Department of Advanced Information Technology, Graduate School of Information Science and Electrical Engineering, Kyushu University

Abstract

In recent years technology has implanted itself into our society and has rapidly become a crucial component. For many it has become an indispensable part of their daily routine, having a mobile phone within reach throughout the day has become common for many, while for others remotely working from home has become the norm. At the same time, the level to which society has become dependent on technology for its normal functioning has brought with it an abundance of ways to exploit those functions.

Because of these threats, it is no surprise that the importance of securing networks, devices, and systems has become a primary concern across the globe. Although a great amount of effort has been put into researching security methods, because of the speed at which new technologies are developed and implemented, it is impossible to completely secure and eliminate all vulnerabilities. Because of this, we find ourselves in a constant cycle to develop new solutions to stop the next wave of threats that appear.

The purpose of this research is to investigate and develop a practical framework and system that can take these aspects into account and help push research into a direction in which we can break the cycle of having to constantly be one step behind attackers. The key to achieving this and getting in front of attackers is to support the existing security infrastructure with a proactive component where potential threats are actively searched for and investigated. There is a deep need to investigate how a system can be developed and implemented which can complement the current security infrastructure to improve cyber defense in the area of pre-attack detection.

In our first contribution, we present a framework for a real-time system where we apply natural language processing techniques to Twitter posts allowing us to classify and process large amounts of data to generate relevant cyber situational awareness information. This framework addresses crucial components in this pipeline such as the filtering and integration of relevant cybersecurity data from a larger data stream, and the integration of new and emerging threat terms into our dataset to keep system filters relevant and up to date. This framework also takes advantage of the inherent context contained in textual data by leveraging sentiment analysis techniques to gain greater insights into the importance of captured threats. By tracking tweets containing security terms that have a negative sentiment we can distinguish pronounced peaks that correlate to security events which can be used by analysts. By separating these streams based on the particular security terms being used we can also gain further insight into the type of attacks such as phishing, spyware, or cross site scripting. In this way, we use information retrieval and text analysis techniques to identify, extract, and analyze emerging cybersecurity topics from the data stream.

Our second contribution builds upon the previous research by delving deeper into the data classification component of the framework. Correctly filtering out non-relevant cybersecurity data from a larger stream of general data is a crucial piece of the system and has a huge impact on this system ’s output. If the data being inputted into the system is of poor quality and contains non-relevant or non-significant data points, we cannot expect a reliable output. Because of this the effective filtering and mining of data can be the most important component of this system. To address this, we present an original method called multi-layer keyword filtering to classify cybersecurity-related text data. This method builds upon traditional keyword filtering methods by integrating a word embedding-based filtering layer which is used to limit false-positives from being retrieved from data streams. This method achieves an F1-Score of 0.99 on various subsets of our dataset and does especially well with short unstructured textual data. This can be attributed in part to the addition of our associated word list which gave up to a 7.8% increase in F1-Score. This is further expanded by introducing a clustering function on post classified data. By clustering our classified data we were able to identify non-significant data points such as advertising and marketing posts which can be removed to create a higher quality data stream. By implementing this method as a whole, the quality of data inputted into a cyber threat intelligence system is increased and hence the reliability of its output improved.

Our third contribution looks at how our solution can be implemented in a production environment across organizations that either do not have the expertise, manpower, or budget to implement traditional threat intelligence systems. Threat intelligence systems have traditionally been adopted by organizations that have the required expertise and budget to implement them. This is an issue since it has left smaller businesses and individuals without this valuable resource. Because of this, we analyze the applicability of our system against other statistical models based on key features such as processing speed, architecture, scalability, and expertise. Based on these features we have been able to outline various scenarios where we can identify the ideal implementations of our system for organizations at various levels. Based on our research the overall ideal implementation of our threat intelligence system uses ensemble learning with our multi-layer keyword filtering method. By using an ensemble method, we were able to achieve an F1-Score of 0.9652 with a processing time of 275ms achieving a good trade off between speed and classification. By doing this we ensure it is possible for all organizations to implement a cyber threat intelligence system by lowering the barrier of entry and hence increasing the level of security across all organizations.

Through the development and implementation of this research, we have found that open-source intelligence platforms such as social media, forums, and other sources are a rich source of various types of threat intelligence information. Using our method, high-quality data can be effectively mined from these sources and used within a cyber threat intelligence system to create indicators that can assist in achieving a better understanding of the cyber threat landscape. This information can subsequently be used to actively implement defense measures in a network to decrease the risk of being compromised. This research contributes to the existing cybersecurity knowledge base by investigating, developing, and implementing a systems-based solution which assists in proactive cyber defense using cyber threat intelligence. By investigating and creating this system we have taken a step forward in finding a solution to an issue that is crucial to the current and future defense of our networks and society as a whole.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years technology has implanted itself into our society and has rapidly become a crucial component. For many, it has become an indispensable part of their daily routine. Having a mobile phone within the same room for the majority of time in the day [34] has become common, while for others remotely working from home has become the norm [19] with their internet connection allowing them access to their workplace and to complete their daily tasks. This brings with it new choices as to how to prioritize the various aspects of this new technological ecosystem. There are many aspects to consider for individuals and businesses alike such as public and private facing infrastructure allowing access to employees, digital privacy, compliance, and many others. One of the most important of these aspects is cybersecurity. There has been a rise in the financial impact of cyber threats hitting various industries around the world and many of these industries are not prepared [50]. Some of the issues preventing them from being prepared are a lack of skilled cybersecurity professionals and an absence of useful security orchestration automation and response software (SOAR) [31].

The level to which society has become dependent on technology for its normal functioning has brought with it an abundance of ways to exploit those functions [125]. Regrettably, malicious actors exist who are eager and capable of exploiting these vulnerabilities who do not consider the consequences of their actions. Because of these threats, it is no surprise that the importance of securing networks, devices, and systems has become a primary concern of national security for many countries and businesses [44]. Being able to detect cyber threats rapidly is a key element to keeping a system safe, this is important for all attacks but is especially the case with the rise of disruptive attacks such as ransomware and wipers which can encrypt and erase systems in a short period [76]. The rise in this type of disruptive attack shows that attackers continue to evolve their tools to achieve their goals whether they be financial, espionage or simply to cause damage [94]. Although a great amount of effort has been put into researching new and innovative security methods, because of the speed at which new technologies are developed and implemented [124], it is impossible to completely secure and eliminate all vulnerabilities. Because of this, we find ourselves in a constant cycle to develop new systems, techniques, and processes to stop the next wave of threats that appear.

The purpose of this research is to investigate and develop a system that can take these aspects into account and help push research into a direction in which we can break the cycle of having to constantly be one step behind attackers. The key to achieving this and getting in front of attackers is to support the existing security infrastructure by adding a proactive component where potential threats are actively searched for and investigated.

In the past best practices in cybersecurity often took the form of network security devices and host-based security systems. These systems continue to be the cornerstone of today's cybersecurity technologies and will remain to be crucial to cyber defense especially in a defence in depth (DiD) model [9]. However, these devices have been developed for a specific purpose that is inherently reactionary. This means that to move towards a proactive method of cyber defense we must build upon our existing security infrastructure with systems and devices that address the existing security gaps and complement existing technologies in both the technical and human areas [131].

Indicators that can give us this insight can range from technical values such as IP addresses, domains, and port numbers. These indicators are commonly used in traditional security devices and lend themselves to a reactionary security posture [104]. But apart from technical indicators, news and social media posts can provide contextual data that can contain valuable information [123]. Traditionally, it was common for analysts to become aware of new threats by reading security websites and blogs [41], this is a slow and inefficient method that has many issues. However this method is still used in a modern day form, it is common for analysts to use social media to gain this information about new attacks and threats. Compared to traditional news, social media can act as a real-time aggregator for various media sources allowing analysts to obtain new information quickly. But due to the sheer number of articles, posts, proof of concepts (POC), and other information available, it is time-consuming to go through and can inhibit productivity through overuse [132].

There is a deep need to investigate how a system can be developed and implemented which can complement the current security infrastructure to improve cyber defense in the area of pre-attack detection [104]. Since the threat which we are trying to nullify has emerged due to the increasing ingenuity and new threats developed by attackers it is only right that we also use emerging technologies to protect against them [114][42]. The increased applicability of machine learning models in recent years has meant many tasks that were previously not possible can be reevaluated for new use cases. The explosion of data mining and machine learning methods presents us with techniques to efficiently classify, process, and analyze large amounts of data and present it to users [24]. Utilizing these methods we can automate the task of reading large amounts of cybersecurity post data and process that data to extract relationships and patterns within it. This allows users to cover a large amount of information rapidly and hence allows analysts to use their time more efficiently to mitigate these threats.

In this research, we present a framework for a real-time system where we apply natural language processing techniques to Twitter data allowing us to classify and process large amounts of data to generate relevant cyber situational awareness information. This framework addresses crucial issues in this pipeline such as the filtering and integration of relevant cybersecurity data from a larger data stream. We also consider the integration of new and emerging threat terms into our dataset to keep system filters relevant and up to date over time. This framework takes advantage of the inherent context contained in textual data by leveraging sentiment analysis techniques to gain greater insights into the importance of captured threats. By tracking tweets containing security terms that have a negative sentiment we can distinguish pronounced peaks that correlate to security events which can be used by analysts. By separating these streams based on the particular security terms being used we can also gain further insight into the type of attack used, such as phishing, spyware, or malware. By taking this one step further we can associate this information with organization names contained within the data to obtain a view of the risk profile for a particular company or sector at a given time. In this way, we use information retrieval and text analysis techniques to identify, extract, and analyze emerging cybersecurity topics from the data stream giving us insights into the data which could not be achieved through manual investigation. The aim of creating this framework is to present the groundwork for a cyber threat intelligence (CTI) system that can be used to assist both humans and devices to have a greater view of the cyber threat landscape at any given point and assist in preemptively anticipating possible threats and quickly sharing news of attacks which are occurring.

During the testing phase of our framework, we became more acutely aware of the importance of having quality data input into this type of system and how substandard data can negatively affect output [112]. This can be expressed through the analogy of a skilled chef who has bad ingredients. Regardless of the skill of the chef, without good ingredients, it will be difficult to produce a high-quality meal. Similarly, this is the case with our system, if our system does not have quality input data it will not produce reliable output that accurately reflects threats, giving no reason to use it. Because of this, we must ensure that the data which is input into our system is accurate and relevant.

Therefore, to enhance our framework we expand upon the data classification component. Correctly filtering out non-relevant cybersecurity data from a larger stream of general data is a crucial piece of the system and has a significant impact on this system's output. To address this, we present an original method called the multi-layer keyword filtering method to classify cybersecurity-related text data. This method builds upon traditional keyword filtering methods by integrating a word embedding-based filtering layer which is used to limit false-positives from being retrieved from data streams. Traditional keyword filtering techniques are used in various areas like search engines, security controls, and other types of information retrieval tasks. Although this technique has various benefits it can produce false-positives because it can not distinguish between the various contexts used in certain terms. This is especially an issue in the cybersecurity domain since there is a large amount of "jargon" which can be used in various contexts. To disambiguate these terms we look at the context in which the term is used by scanning the other terms within the post for common terms that are used in a cybersecurity context. This is achieved by measuring the cosine distance between the term in question and other terms that or commonly used with it within a cybersecurity context. Using this method within our system achieves an F1-Score of 0.99 on various subsets of our data and does especially well with short unstructured text data such as Twitter data. This can be attributed in part to the addition of our associated word list which gives up to a 7.8% increase in F1-Score.

By utilizing this method we can improve the quality of data that is used and stored in our system by removing non-relevant data that is not related to cybersecurity. To further improve the quality of data and of our output we also consider the significance of data that we use within the cybersecurity domain. Within the domain of cybersecurity posts, there is a large range of topics that are discussed [64], these topics are not limited to discussion about threats. This can include personal discussions, discussions about conferences, certifications, new products, and much more. This means that within the scope of cybersecurity posts we have a hierarchy of significance with some posts contributing to the detection and discovery of attacks and some posts which have very little significance being applied to this task. Posts related to marketing and advertising is one particular example which does not contribute to the detection of new or current threats and can even harm detection by skewing the results of our system. Therefore it is logical that we further expand this component to identify and remove this type of data to consider data quality considerations and improve the quality of our output [46].

We do this by introducing an unsupervised clustering algorithm on post classified data. By doing this we can identify particular topics within each cluster and apply significance to that data. For example clusters such as advertising and marketing posts can be removed to create a higher quality data stream. By implementing this method, the quality of data inputted into a CTI system is increased and hence the reliability of its output improved. This data can then be stored in a relational database which is accessed by a front-end dashboard. The dashboard can be used to provide real-time analytics of our data which users can use to gain insights that could not be obtained by manually scouring those sites. Information retrieval and computational linguistics methods can be used to aggregate our data and provide a summary of the most important cybersecurity topics for a specified period which can interactively be filtered by n-gram length and date to gain further context. Since this data is made up of social media posts we can allow users to investigate further into particular topics by providing access to the original Twitter posts allowing users to analyze the details of each post.

This research does not only consider the technical factors involved in creating a threat intelligence system. One of the greatest obstacles to the adoption of new tools and technologies comes down to factors such as a lack of expertise or training on new tools which means that even though an organization may have access to a threat intelligence system they do not integrate that data into their analysis. In other small businesses and individual cases, they may question if they have adequate infrastructure, budget, and personnel to implement and maintain a computer based information system [12]. To address these questions we explore and outline the various implementation solutions which are possible with our implementation of a CTI system. We analyze the applicability of our system against various scenarios based on key features such as processing speed, architecture, scalability, and implementation expertise. Based on these features we have been able to outline various use cases where we can identify the ideal implementations of our system for organizations at all levels. Based on our research the overall ideal implementation of our threat intelligence system uses an ensemble learning method including our multi-layer keyword filtering method. By using an ensemble method we were able to achieve a high F1-Score of 0.9652 with a processing time of 275ms achieving a good tradeoff between speed and classification. This particular implementation is a good trade off between speed and classification but other implementations can be used to either maximize classification or maximize efficiency. We consider the scalability of this type of system based on the data mining policies and the type of architecture needed.

The output of this system is designed to be able to be used by both knowledgeable users such as security analysts and also less technical users. Security issues are not limited to companies, individuals also find themselves being increasingly targeted by various attack vectors such as online scams, phishing [65], ransomware [7], and magecart style attacks which target common platforms such as social media, e-commerce, IT service providers and mobile devices [72][76]. Finding ways to intuitively assist users at all levels can help increase threat awareness and prevent the impact of various cyber threats [109].

Through the development and implementation of this research, we have found that open-source intelligence platforms such as social media, forums, and other sources are a rich source of various types of threat intelligence information. Using our method, high-quality data can be effectively mined from these sources and used within a CTI system to create indicators that can assist in getting a better understanding of the cyber threat landscape. This information can subsequently be used to actively implement both defensive and offensive defense measures throughout a company's network to lower their risk of being compromised.

By investigating and creating this system we have taken a step forward in finding an answer to an issue that is crucial to the current and future defense of our networks and society as a whole. Open-source information in the form of social media and other news outlets has provided us with a wealth of information that can assist with threats.

But the amount of information has grown to become an issue and analysts either have to know various technologies to access this data or manually extract actionable points from it which increases the learning curve and effort for analysts. It also increases time for investigations, and limits access to this data to a select few. Through this research, we present a framework for organizations at all levels to implement cybersecurity analytics based on big data which gives insights into real-time cybersecurity topics. We lessen the learning curve for less equipped users and allow experienced users to more efficiently and rapidly perform common tasks that would often be done manually. In this way, this research contributes to the existing cybersecurity knowledge base by investigating, developing, and implementing a systems-based solution that assists in proactive cyber defense through the use of CTI.

1.1 Background

The first wide-area packet switching network was called Advanced Research Projects Agency Network (ARPANET), this network laid the foundation for the internet and implemented protocols which are still used today [82][26]. The creators at the time had no way of knowing the importance this network held, nor could they foresee how important the need for security in the protocols which made it up. Hence many of the original internet technologies were built without security in mind [18]. Over time various iterations of TCP/IP were implemented into the network and access was expanded. Eventually, partnerships with telecommunications and computer companies brought the widespread commercialization of the network which now spanned the world and had been named the internet (international network) [134].

The lack of ingrained security has led to a whole industry of devices, products, and services aimed at securing our networks and hosts from threats. Firewalls, antivirus software, intrusion detection systems, and countless security protocols were all developed to protect our networks. These products were created to attend to specific security goals and have subsequently evolved to do a multitude of security tasks. For example, one of the most iconic security devices that exist is the firewall and was designed to filter packets that flow through it [102]. The first firewall examples are generally recognized as work done by Digital Equipment Company (DEC) in the late 1980s. During that time firewalls were essentially packet filters that were able to drop incoming or outgoing packets that passed through the device. These packets could be identified based on Internet Protocol (IP) address, the protocol they used, and the source and destination ports [27]. During this period in time, the internet did not contain the number of protocols and services which have come to be developed in today 's network [5][45]. Therefore this limited packet filtering based on port numbers for common services provided good security, and allowed a company to control incoming and outgoing traffic from their network.

Although these devices can filter traffic based on IP, port, and protocol they know nothing of the data which is being carried within a packet's data field. Therefore application-level attacks such as cross-site scripting (XSS) and SQL (Structured Query Language) injection cannot be detected by only using a firewall. To fill these gaps more devices were created, the intrusion detection system (IDS) monitors networks and systems for suspicious activity [55]. It can report suspicious traffic based on signatures that have previously been identified as malicious but this type of system cannot detect new attacks that have not previously been seen and hashed. To find known attacks, anomaly-based systems were introduced to counteract the development of new strains of malicious software. Initially, anomaly-based intrusion detection systems were based on rules rather than signatures. By monitoring normal traffic patterns a company could create a baseline to compare against and detect abnormal scenarios [60]. This system can then be used to generate alerts depending on how much it differs from the recorded baseline. Although these early systems helped with detecting new threats, this rule-based system has the potential to generate false alarms due to its rule-based approach which can lead to alert fatigue [14].

Unknowingly at the time, these devices fomented organizations into having a reactionary security posture by alerting the organization when an attack is already in progress or has already finished. This is principally in cases where some type of sensor or device is used to record traffic within the network and generate an alert based on that recorded data. Although being alerted of an attack in any way is beneficial in the fact that you know an attack is happening or has occurred, it often does not allow analysts and security engineers adequate time to mitigate, or to meaningfully understand what is happening [4]. In the case of ransomware and wiper attacks finding out that an attack is occurring whilst it is happening is too late, and can mean the loss of infrastructure, data, and large costs. This leads to the question of how we can become aware of threats before they occur so that companies can prepare better to defend themselves.

Implementing proactive defense has traditionally been a difficult task, in legacy networks, there was no practical way to anticipate attacks with the technology available at the time. Today's internet landscape has evolved exponentially and the technologies which have been developed in other areas have advanced to a point where they can be applied to this problem. Two specific areas that can greatly contribute to fixing this issue are machine learning (ML) [85] and big data [110]. These two technologies have contributed to each other's success with the boom in big data providing valuable opportunities for training sets that have helped supervised machine learning models gain better results in a range of different domains [141].

Due to this increase in content platforms and increased accessibility to technologies such as ML and data mining which facilitate the capturing, processing, and classification of this data, proactive defense solutions have now become more practical. Even though this may seem that it has happened quickly from an outside perspective there is a history of research being done into how security applications can use these techniques to enhance security.

Initially researchers looked at integrating machine learning and data mining into existing security appliances using common indicators such as IP, port numbers, and protocol types. Early applications worked on classifying traffic using clustering [80] and supervised learning methods [107][89]. As mentioned supervised learning methods require training data to be effective and hence various datasets were released to facilitate research into building models and systems which could detect cyber attacks [67][32][126][90]. These datasets are largely created using network traffic data. Although this information is important and has its place in network defense. To promote a more proactive defense we must leverage more contextual data through CTI.

CTI refers to information about threats and malicious actors that allows for the prevention and mitigation of attacks. This can come in the form of actionable information about cyber threats such as malware, DDoS, phishing campaigns, and many more. By having greater intelligence on these threats such as release dates of new versions, authors, signatures, and other indicators, we can apply better decision making. Threat intelligence can provide information which contains context and directly actionable advice to help make complex decisions about securing the network or a particular service.

Threat intelligence can be broken down into three categories, tactical/technical, strategic, and operational. Tactical intelligence often refers to specific details on malicious actors such as specific technical indicators and tactics techniques and procedures (TTP). Strategic intelligence generally refers to high-level organizational strategy including the high-level decisions that must be made and how an attack can affect an organization's wider functions. Finally, Operational intelligence refers to actionable information about attacks and the motivation and capabilities of the attackers. Through these information types, the generation of CTI can allow an organization to develop a proactive cybersecurity defense that allows it to not only be reactionary but also prepare for threats in advance. To be able to generate this intelligence we first require a data source that contains relevant information within it. Ideally, a CTI system would draw from a range of sources to provide a well-rounded view of threats in the landscape. There are various sources that can be used to implement a CTI system, some examples of these are technical indicators of compromise like IP's, hashes, domain names, and other signatures. Specific log data and other client-based data generated from SIEM's and SOAR appliances can also be used. The issue with these data sources is that they are generally closed sources meaning they are not publicly available and are often confined to local sources which reinforce a fairly reactionary stance.

On the other hand using open-source intelligence (OSINT) sources such as social networking services (SNS), open and closed deep web sources, and traditional news data we can start to move away from a reactionary defense posture towards a more proactive and predictive one. The use of social media has become widespread throughout all social circles including the cybersecurity community. This includes being used by malicious actors to communicate between themselves, and as a platform to sell malware and attract customers. Social media is also often used by hacktivists which may give indications of their intended actions towards specific targets beforehand [56]. Because of the amount of data which is generated by social media and the amount of context which it provides, it is without a doubt one of the best repositories of data for generating CTI.

Other systems that can take advantage of these sources have been researched previously. For example, Sabottke et al. [108] explore a method that provides early exploit detection based on Twitter data such as keywords, retweets, and replies. They utilize supervised machine learning using a Support Vector Machine classifier (SVM) that is trained on tweets containing the keyword "CVE".

Common vulnerability and exposures (CVE) is a reference method for vulnerabilities that allocates a CVE id and have a description of a vulnerability. Tracking CVE mentions on social media such as Twitter is a popular way to mine cybersecurity data and has been widely utilized [62][51][28] for tracking, predicting, and monitoring threats. Social media such as Twitter and Reddit are shown to have mentions of CVE's up to a year before their public disclosure date allowing security teams to better prepare for threats. But out of the 45,450 public exploits that are available through Exploit DB only 26% have mapped CVE numbers, and upto 80% of public exploits are published before the accompanying CVE is released [29]. This shows that while tracking CVE number mentions is important, limiting systems to this data gives it a narrower scope. Although social media hosts a wide range of cybersecurity information of varying types. CTI systems benefit from obtaining a wide range of data to get the best insight into the current threat landscape.

Another source of threat intelligence is darknet/darkweb forums and markets. This includes sites that are hosted on anonymity networks such as The Onion Router (TOR) [36] and the Invisible Internet Project (I2P) [137]. These sources can contain valuable insights into the current trends of malware, data leaks, and tools that are popular in the underground community. By tracking this information it is possible to get a good understanding of emerging threats and exploits that are developed by individuals and traded for large amounts of money. This is an area that is being researched heavily both in academia and in private companies. Almukaynizi et al. [11] present Darkmention a system that mines these forums and generates alerts to security operations centers (SOC) before attacks occur. Darknet forum and marketplace mining for threat intelligence is also a popular area of research [33][38][113] and similarly attempts to generate alerts that can inform security analysts about threats before they happen.

Obtaining this data can often be difficult since it requires custom data mining scripts that can run on a particular anonymous networks [36]. Also, many forums are invite-only and are very cautious in regards to allowing access to new members. While marketplaces are easier to access they often contain lower-level exploits and generally a lower amount of malicious data about items sold [93]. These factors contribute to a scope issue, and by only mining darknet sources for data we are also limiting usable data for a general CTI system.

As has been shown there has been increased interest in exploring the area of CTI systems and the generation of threat intelligence in recent years due to the great potential benefits which can come from this type of intelligence. Various research and private systems have been developed and proposed to help companies fill the gap in their defenses. There has been a trend moving towards the automation of various security tasks by developing various services and systems to tackle each of these tasks. This in itself has created its own problem of device saturation.

Various technologies are required in today 's environment to keep track of the various data flows which need to be kept secure, which in itself has caused issues. This increase in tools, systems, frameworks, and services which are available has become overwhelming for analysts. The amount of alert generating tools has snowballed to a point where many alerts are not able to be investigated [2] by analysts and to make matters worse after investigating alerts they are often found to be false-positives meaning time has been wasted investigating a threat which does not exist. This is referred to generally as alert fatigue and/or alert overload. In the worst-case scenarios alerts are ignored [79] or key tool features turned off [3] removing the benefits which can be achieved from these tools. This is a very dangerous position to be in for companies. These technologies are very powerful and are a key tool to help analysts mitigate attacks. If an analyst can not trust the alerts being generated by a particular system and does not use it, there is an overall failure. The company is paying for a product which is not being used and analysts are not able to reap the benefits of products which should be making their jobs more efficient.

Because of this, it is important to use machine learning and data mining as tools to empower analysts by enhancing their existing skills and improving efficiency in time-consuming tasks. Ideally, a system should enhance an analyst 's capabilities and provide fewer alerts that are of higher quality to ensure productivity. Having an environment with frequent alerts, where analysts can not investigate any leads does not improve efficiency for any of the parties involved and has to be avoided.

In this section we have shown a brief history of the path which has brought us to the point which we now find ourselves at. This path has fostered the ideas for the research we have created and our methodologies for creating a CTI system. Our research has looked to learn from the past and use it to develop a framework and system that aggregates and analyses open-source cybersecurity information. This information can then be processed, analyzed, and visualized for better understanding from an analyst. Rather than generate security alerts that can contribute to alert fatigue we provide a method that enables users to investigate cybersecurity topics such as threats and subsequently triage those threats based on the collected data. We consider a wide scope of open-source data since it is a more effective way to aggregate a wide range of cybersecurity news, and has the possibility to contain a large number of security vulnerability mentions compared to using a single particular platform.

1.2 Related Work

In this section, we give an overview of the existing academic research and studies which have been conducted in our area. We look at the various aspects which make up our research as well as the research as a whole and analyze other works within the same domain. This is done with the goal of situating our work and showing its position within the existing literature.

1.2.1 Open Source Data Platforms

There are various researches that have looked at data mining Twitter and other open sources for valuable information. Here we outline some researches which have shown the value that these sources have and give insights into ways they can be used. In [15] Twitter is used as the data source to investigate the connection between malicious actors who focus on website defacement and how Twitter data can be used to understand their behavior. This was done using a list of Twitter accounts that were linked to malicious actors.

In Islam et al. [54] they look at modeling the ecosystem of malware authors by analyzing the online repository site GitHub along with security forums. According to this research, the number of new malware authors on GitHub is roughly tripling every two years. Along with this increase, many malicious actors are creating a brand for themselves by marketing their open-source GitHub software through other security websites and maintaining their username across multiple platforms. This activity speaks to the importance of monitoring various platforms within the security ecosystem and getting a scope of data that is able to correlate these points. By tracking software repositories it is possible to find the source of a piece of malware which is much more valuable than identifying a user who simply copied it. This scenario is common, with malware repositories being 4 times more likely to be forked compared to normal GitHub repositories. This speaks to the higher collaboration and technical literacy within the hacker community. They find that a small number of authors drive the community and are part of a high influence group. By being able to identify this group this system can track released malware by the authors and users who copy this malware. This work aims to characterize the hacker online ecosystem in relation to open software repositories and has applicability to malware attribution and tracking. In our case, we are trying to achieve a larger holistic view of the ecosystem rather than specifically honing in on malware. Although some users are willing to post their malware on an open-source repository like GitHub and attach their online screen name to it, these can easily be migrated to private repositories that can not be accessed.

Horawalavithana et al. [51] provide a quantitative analysis of vulnerability information on Reddit, Twitter, and GitHub to find the relationship between vulnerability mentions, platforms, and how social media discussions drive development on GitHub. A dataset based on common vulnerability identifier (CVE) mentions and GitHub accounts that have a CVE identifier in the repository description was used for training machine learning models to predict the popularity of certain GitHub repositories based on Reddit and Twitter mentions. When looking at CVE ID's they are mentioned more on Twitter than on Reddit. In our research, we test various platforms including Twitter and Reddit to get the greatest view possible of all information. Therefore it is valuable that [51] was able to show that on both platforms CVE identifiers were discussed before their public disclosure date. Both platforms had mentions of vulnerabilities more than a year before their disclosure date. This shows the importance of not only mining these open sources for data but also the importance of mining various sources within this sphere which is a factor we consider in our research. Having said that it is important to understand the characteristics and features of each platform. This can influence the data mining method used and the weighting of data points within an algorithm. This study found that the majority of CVE mentions found on Reddit happen before public disclosure while on Twitter the majority happen after public disclosure.

CVE information is one particular indicator that may appear on social media and forum sites. CVE numbers are a good indicator since they provide further context on the vulnerability such as time of discovery and general information and severity. Using a single indicator solely within a CTI system is generally not optimal since you are limiting the scope to a very specific aspect. CVE numbers are often not known by general users and tend to be used by people who are security conscious or in the cybersecurity industry. In this way, it is useful to gain insights into more knowledgeable discussions but less so for the general public who might make a comment about being attacked who are not aware of the CVE id.

Kim et al. [59] present CyTIME, a CTI management framework for managing CTI data. This framework uses CTI data repositories in the form of open Trusted Automated Exchange of Indicator Information (TAXII) Servers to acquire data and then convert this data into a JSON standard format for CTI known as Structured Threat Information eXpression (STIX). TAXII enables the secure sharing of CTI information in the STIX format which is a standardized way to describe cyber threats through their domain objects such as malware type, identity, and campaign. STIX and TAXII provide a standard way for organizations and users to securely exchange and store CTI. This information can be stored on an open or closed TAXII server which can be accessed using a TAXII client. These services have been crucial in promoting the sharing of CTI between government, commercial suppliers, and other organizations to increase awareness of attacks as soon as they occur. Unlike our previous examples, TAXII servers can be used as a type of CTI feed that can be used to download timely information about various threats. In this scenario, rather than using a source first hand to develop an indicator, they are relying on data collected and structured by other parties. In the case of open TAXII servers, the origins of the data may be unknown and therefore its trustworthiness can be questioned. This type of open-source for CTI differs from using other open sources such as social media since in this case the indicator has already been created and is ready to be analyzed and consumed, this makes the process simpler but removes the in-depth knowledge of the creation of the data itself. In our case we take an end-to-end view by looking at the whole process of developing and distributing indicators. Having said this formatting our data in a STIX format and making it available on a TAXII server can increase the effectiveness and reach of our system.

1.2.2 Dark Web Social Network Data Mining

When we talk about social networking services we often think of the most common platforms which are available through the clear web such as Facebook, Twitter, Instagram, and many others. These are the most widely used and best known social media platforms but do not account for the whole market.

One particularly important area of the social network ecosystem when considering the cybersecurity community are platforms which are hosted on anonymity networks such as The Onion Router service. Although it takes a somewhat different form to its clearnet counterparts the network of underground forums, marketplaces, and dark web communities can provide a treasure trove of intelligence on cybersecurity-related threats such as malware, carding forums, data leaks, and much more. The type of data and interactions of users on darknet markets and forums is an area that is well studied.

Nunes et al. [93] develop a CTI system to gather information from hacker forums and marketplaces on the dark web. This system identifies relevant data from these sources such as products and posts containing information in relation to malicious threats. They found that similar to clearnet social networking services it is common for malicious actors to belong to multiple sites within the ecosystem. By doing this they are able to advertise their product to the widest audience. This shows that in the same way that a wide scope is necessary to view all possible threats on clearnet services, the same case is present on darknet services. The value of the data which can be found in these sources is shown also. In this research, they were able to detect 16-cases of zero-day exploits in a 4 week period. Zero-day vulnerabilities are very valuable to attackers since there is no public knowledge of the vulnerability and therefore it can be easier to exploit. Having knowledge of zero-day vulnerabilities is valuable to a CTI system by being able to convey this knowledge of new threats to the analysts which they can begin to control. Also, mining dark web sources allow us to integrate discussions which are not available on clearnet social networking sites which can contain valuable information about threats.

In Marin et al. [78] they study dark web sources from a hacker centric perspective to identify key figures within the ecosystem. They found that the community which makes up this environment consists largely of unskilled users and users with a passing interest in hacking. This, therefore, makes the identification of key hackers a more difficult problem. In terms of retrieving posts from these platforms for a CTI scenario, it increases the number of low-level posts which although relevant are less valuable. Although these posts should not be discarded altogether, by being able to identify key figures in the scene it is possible to get more impactful data. In this research, they are able to use various features derived from content and community hierarchy measures to identify key hackers and show that this method can be used across forums that contain the same features. The hacking scene has a hierarchical structure where skilled hackers hold a high reputation based on their achievements and exploits. These highly skilled users often make up a fraction of the entire community but are very influential within it. Its possible to track users across sites by the fact that users with high reputations often want to be recognized, because of this it is more likely that they will keep the same username across multiple platforms. Although, this tracking method is obviously easily bypassed with the user creating a different account under a new username. In this research, they consider tracking the key hackers across forums by using reputation scores on individual forums to identify key hackers. This information is very useful in the identification of new malicious software types. By using this data within a system we can begin generating indicators of compromise (IoC) for malware before it is widely used.

Liu et al. [68] identify, collect, and monitor personally identifiable information (PII) across the dark web and surface web. RSS feeds are used as an indicator of data breaches that are likely to contain PII information. Based on these alerts a dark web crawler is used to retrieve HTML content such as name, city, state, etc. To enrich this data, queries are conducted on the surface web-based on the PII data found on the dark web. Common sources such as LinkedIn and other sites can be used to correlate PII such as names and email addresses to gather further PII on the targets. This shows how CTI systems can use both the dark web and surface web together as open sources to provide a user with an idea of the level of risk they are under depending on the amount of publicly available personal information. Similarly, the service ”Have I been pwned” [53] allows users to search for their email address in recent PII breaches letting them know if they should change their password. In this way, we can see how various open sources across all platforms and types can be a useful source of OSINT and can then go on to be used within a CTI system.

1.2.3 CTI Frameworks, Systems & Tools

We have shown the various open sources such as social media, the dark web, forums, and others that contain a large amount of data which we can leverage to generate an alert or other type of notification to assist with security defense. Because of the depth of data that is available across these sources, frameworks, systems, and tools have been researched and developed. They cater to a number of different areas that apply a specific method or indicator and judge its value within the cybersecurity domain. Within this section, we look at some of these researches, especially those which share some conceptual or technical aspect with our implementation of a CTI system. Some of the common components of these types of systems that are used are data retrieval, classification, processing, and visualization. Although these components are common through many CTI systems their implementation can vary depending on the use case, goal, and intended audience. We review different systems and their implementation of these components to give greater context on their implementations and also why we chose to use our particular implementation in our design.

Mittal et al. [88] present CyberTwitter as a framework to analyze tweets and generate alerts. This system uses a security vulnerability concept extractor to extract core concepts of an attack such as hardware, software, and other attack concepts. These concepts can then be linked to external entities using existing knowledge bases to gain more context. Based on the retrieved information and the time of the attack a security alert will be generated if it is deemed to be a threat. Le Sceller et al. [63] use Twitter data to identify and monitor cybersecurity events. A list of 200 cybersecurity-related keywords is used to filter the Twitter stream and narrow it down to more relevant data. This list is able to update itself to include new terms that are relevant by using a combination of word embeddings to generate a list of related terms and then ranking those terms based on Term Frequency Inverse Document Frequency (TFIDF). This system also uses a keyword blacklist to lower false-positives by discarding tweets that contain a term from the blacklist. This blacklist contains ambiguous phrases which could be judged as security-related when using the keyword filtering method such as ”hacked the door down”. The keyword filter method is prone to false-positives and therefore it is good to include a method to handle ambiguous terms. Using a blacklist method to achieve this where exact terms need to be matched can have difficulties. The updating and maintenance of such a blacklist would be laborious due to the number of attacks and having to include new attacks into the filter. The size of this type of blacklist would also be very large if it were to cover all possible terms. In our research, we implement our MLKF method that can be used for the classification of cybersecurity-related tweets which also considers the ambiguity of terms. Rather than use a blacklist we use a dynamically generated whitelist which generates a list of common terms often used in conjunction with the seed term. This reduces the need to have a large blacklist which needs to be manually maintained and updated.

Zong et al. [144] use Natural Language Processing (NLP) to analyze the severity of vulnerabilities through Twitter and link them to user accounts and the National Vulnerability Database (NVD). In this system, Tweets were first captured using a keyword filter and then manually annotated for threat relevance and severity. Using this dataset they then are able to train a machine learning model to estimate the likelihood that a tweet is referring to a cybersecurity threat and its severity. Similarly to our research, the models are trained using bag of n-grams features and word embeddings. This method provides information that can then be used as an early measure of a vulnerability. This research initially uses a keyword filter to mine cybersecurity-related posts. This filter is based on two words and then manually annotated for relevance. Within this process, 2543 tweets out of 6000 or 42.38% were classified as relevant. This shows the high amount of false-positives that can occur using just a keyword filtering method. In this research, they were able to mitigate this aspect using manual annotation, but when dealing with large amounts of data which is often required for the latest applications of ML models manual annotation is not an option. This shows an area where the MLKF method which we present as part of this research can be used within the academic research systems and also in industry to automatically remove the number of false-positives and enhance the quality of datasets.

Dionisio et al. [37] present an end to end neural network-based pipeline which uses a curated stream of Twitter data employing two classification models to generate a security event for suspicious data points. Firstly they curate tweets based on a keyword filter and input this stream into a binary classifier which uses a convolutional neural network (CNN) classifier to annotate the post based on its relevance. If the data is deemed relevant this is pushed on to the next stage which is a natural entity recognition (NER) model that labels valuable security-related entities. If some threat is found through this method the information produced can be used to generate a security alert or to enrich an existing IoC database. Using NER is a useful method to distinguish particular entities within a post such as a tweet. In our existing research, we use a static list to identify companies that we would like to track through our dataset. Although this is adequate for situations where you have a set amount of entities which you would like to track, in cases where a system wants to dynamically track targeted entities as they rise and fall it requires a large amount of maintenance. In these cases, a NER model can be considered to detect organization entities in the data to find and aggregate threats specific to those organizations.

Samtani et al. [111] introduce a CTI framework which aims to improve upon the current reactionary stance of security devices by introducing a more proactive approach. This is done in the form of a CTI system that uses SVM and latent dirichlet allocation (LDA) models to collect and analyze data from dark web forums. This information is outputted to a dashboard interface to show assets and threat actors to students and analysts. Asset analysis is done by processing the asset data, thread, and author and identifying the type of programming language used in a post. Once the classification has been performed LDA is used for topic detection based on specified keywords for each topic. The discovered assets such as specific users can be viewed through a dashboard to reveal further information about threats they post and other information. In this research they use assets that are specific to the dark web forums used to great effect, providing a system that can track users and provide valuable information. In our system, we try to not use platform-specific assets for the reason that we want to be able to normalize our system for all types of data sources. The Twitter platform contains various types of assets that can be leveraged to gain more context on threat posts such as users, retweets, followers, and more. These assets are useful but can be difficult to link across platforms in which they may not exist. In the case of news articles many of the assets are not available, also to correlate a username on Twitter to another platform like Reddit can be difficult to do reliably. Because of this in our research, we try to focus more on the data and insights which can be extracted directly from the post itself without correlating it with platform-specific data. By using terms within the body of the data only and using techniques like sentiment analysis which do not rely on platform-specific features we can ensure that our system can be used across any platform in its intended form.

1.3 Cybersecurity Text Classification

In the previous section, we explored the existence of various CTI systems, frameworks, and tools which incorporate various components that are commonly used in these types of scenarios. Each of the common components of this type of system holds value such as data retrieval, filtering, pre-processing, data mining, and information extraction. Even though every component within this system has its use and is important, if the data which is obtained is not of a certain quality the other functions do not have a great impact, since the data outputted is likely to contain errors. Because of this, it is important to consider and test this component independently of a CTI system although keeping in mind that the ultimate aim of the model is to be used for this goal.

NLP looks at the relationship between human language and computers and how this relationship can be exploited to process and understand the patterns in human text data. This is done through various techniques and algorithms, one of those being text classification. Text classification is the process of tagging or labeling a piece of text data with a value that can correspond to various groups such as topic, sentiment, or type. Various researches have been conducted to classify texts from different domains such as politics [103][138], natural disasters [95][25] or virus epidemics [128][57]. Research which focuses on the classification component of text is more common in other domains than it is in cybersecurity. This is due to the emerging nature of cybertext analytics, and the traditional popularity of technical IoCs, along with the fact research in this area usually focuses on the system as a whole rather than the particular component and its efficacy. Having said that, there are examples of research which is solely directed towards the application of NLP on cybersecurity texts. Competitions such as SemEval have promoted more research in this area through shared tasks [99] where participants can choose from a number of tasks related to semantic evaluation and submit their solution and compare it against other teams. One entry for this task among many [73][71] is Manikandan et al. [77] where they present their participation to the SemEval-2018 task 8 which is concerned with semantic extraction from cybersecurity reports using NLP for the classification of sentences for malware and entity prediction. They present a CNN model for malware sentence classification and a Conditional Random Field (CRF) system for malware token prediction.

Shin et al. [117] use a contrastive word embedding model to classify cybersecurity intelligence from Twitter data which employs a technique similar to the one used in our research. By training cybersecurity related and non-cybersecurity related data word embedding models separately, they aim to maximize the difference between embedding models. They use these embeddings as an input to deep learning models such as CNN and long short term memory (LSTM) to classify tweets as cybersecurity related or not. In our case we train our word embedding model on cybersecurity related data as well but rather than using this as an input to a classification model, we create our own MLKF method which uses cosine distance to classify data. We have chosen this route to maintain a lightweight and flexible model which can be easily applied to any domain without the need for extensive deep learning model training.

1.4 Cybersecurity Text Relevance & Credibility

The popularity and deployment of CTI systems are increasing and with more companies utilizing these technologies it means that these systems will begin to be targeted by malicious actors. One particular way AI and ML-based CTI systems can be targeted is through poisoning attacks. In this scenario, an attacker can generate a large number of posts through a bot or human network to influence the output of the system. Using this technique they can affect the output of the system. This will at least cause a distraction and time-consuming event for analysts and at worse could cause changes in the network which could potentially be exploited. There are various techniques to mitigate this risk, some systems filter data based on verified and credible data while others use a credibility score value as a threshold.

In Khurana et al. [58] they use a credibility measurement to determine the credibility of Reddit posts through a reputation score. Some systems ([64][13][15]) handle this by limiting their data based on verified or reliable user accounts but this means severely limiting the scope of the system and limiting its abilities. Here they propose an ensemble SVM and embedding model method which can be used prior to or within the CTI system to classify posts as credible or non-credible data. The dataset was created using Reddit based posts and was manually annotated with a label of credible or not credible. An SVM model was then trained on multiple posts using platform-based features to classify data with a binary credibility label. The SVM and vector embedding models were stacked to produce a score that can be used to judge the overall reliability of the post by a human analyst or device. In our CTI system framework, we consider the legitimacy of data and tackle it using the common technique mentioned of limiting data based on respected user accounts. This is not the optimal technique for a CTI system due to platform-based features such as user accounts not carrying over to every platform and the difficulty in managing each account. The solution proposed in this research looks at effectively adding another component to the CTI system architecture which evaluates data credibility prior to it being inputted into the system. This is a difficult issue to solve and has different characteristics depending on the platform. For instance, in this research, they consider the Reddit platform which has a very different post form and userbase to the Twitter platform. On Twitter between 9% and 15% of Twitter accounts are estimated to be bots [129] which lends itself more to noncredible data when compared to a platform like traditional news articles. The effect of misinformation and credibility is one that needs to be considered especially as the increase in CTI systems becomes more integrated with security operations.

1.5 Topic Clustering

In our research, we concentrate on text-based data in the form of various types of posts, articles, and digital discussions. Although we focus on text-based data, clustering research for security is an area that has been researched traditionally through the clustering of technical indicators. Indicators such as IPs, signatures, hashes, and domains are used often within security devices and in the past have been primary IoCs. K-means clustering on technical indicators has been used for various solutions such as attack clustering [118],[139] and anomaly detection [47] but for our purposes we are interested in text-based data.

Lee et al. [64] present a topic mining service which uses open threat intelligence to find emerging threat topics and present a possible solution. This tool limits its input data based on a set of cybersecurity-domain Twitter accounts considered to be experts within the community. Within this research, they used topic clustering on the corpus of expert tweets to show the range of topics that were discussed within the group. A total of 100 cybersecurity experts posts were analyzed using a clustering algorithm to cluster the data into three groups based on the topic. They were able to find that each cluster outlined a distinct topic. The first cluster was as expected in relation to the security domain. The second cluster was on subjects which are tangential to security in the form of technology interests or conferences. The third cluster largely contained personal discussions about interests such as sports and social activities. This shows the need for identifying and filtering out non-significant topic clusters from our data.

Behzadan et al. [17] develop an open-source web application to assist in the annotation and exploration of Twitter-based OSINT data. This tool is used to propose a method to detect and classify cyber-related tweets. They initially used a keyword filtering method to create a dataset of Tweets which was subsequently manually annotated. These tweets were labeled based on both relevance and threat type with a general/marketing category added for tweets that were related to cybersecurity but were not related to a threat. Two CNN models are used to classify tweet relevance and then for positive results to be classified into the threat category based on the label. In our MLKF+C method, we consider this aspect while trying to improve the quality of our data stream. Using various clustering algorithms we investigate the data groups which are formed from clustering and how we can classify and eliminate clusters that are less significant. In this study, this was handled manually with human annotation at first and then using that data as a training set for a CNN model. On large-scale datasets, it is not practical for manual annotation to be done. We therefore opt for an unsupervised method that can be used effectively for automatic classification.

Alves et al. [13] propose Synapse, a Twitter-based continuous threat monitor that collects, classifies, and clusters tweets. Synapse also uses specific security accounts to filter tweets. To handle out of domain topics it uses a whitelist to drop tweets which do not match the list. This manual method can be useful for a small number of topics but is manually intensive and hard to maintain for CTI systems that aim to get a wide scope of data. Tweets are then classified using an SVM model followed by a clustering phase. CTI systems that use Twitter as their data source have to deal with the possibility of retweets which can mean you have a large amount of the same tweet within the dataset. In this research, they make it easier for analysts to use this system and find the general theme within the tweets by presenting a single ”exemplar” tweet that represents various tweets which are important within the dataset. This is achieved through the use of an adaptation of the Clustream algorithm [6], DynamicClustream. This method implements an online and offline phase which adjusts the number of clusters depending on the appropriateness of the data point to fit existing clusters and dynamically allocating them or creating a new cluster based on this. In a similar vane, Concone et al. [30] develop a method which discards erroneous posts and then clusters the remaining tweets using K-means clustering. An alert subsystem is used to cluster documents and group posts on similar topics. This is done with the aim of more effectively identifying main topics and eventually creating an alert system based on this data.

In these methods, they have taken the route of grouping similar tweets based on the topic using a clustering algorithm. In this process, it is often the case that outlier posts are often discarded meaning that this type of system may miss less popular or obscure tweets that do not fall within a particular group. Within our system we consider both aspects to be important to the effective implementation of a CTI system. The user should be allowed the option to manually go through individual tweets to gain a fine-grained understanding of the data, but should also be able to see a high-level abstraction of the most important topics. Fink et al. [41] discuss the importance of allowing analysts to explore the intricacies of their data and the dangers of having tools which obscure this view. Therefore an effective CTI system should be able to provide both perspectives for its users.

1.5.1 Threat Term Detection and Disambiguation

In Seyler et al. [116] they present a method to disambiguate ”dark jargon”, which they define as ”benign-looking words that have hidden sinister meanings and are used by participants of underground forums”. An example of this could be the term RAT which can refer to a remote access trojan or to the animal or to a person. This method attempts to interpret these terms by using a mapping method based on KL-Divergence. Due to the varying contexts of these terms, it is normal that the ”dark” context of RAT will have a context more similar to malware since a remote access trojan is essentially that. By then taking the word distributions for the normal version of the word and for the dark version of the word, the dark jargon’s meaning can be found by identifying the word with the lowest dissimilarity to the target dark word. In our method, we use word embeddings to determine if a word’s context is within our domain based on the context of a post and the similarity of the words to the terms which are most likely to be contained in that context. For our purposes, we are not trying to identify the meaning of the particular ”dark jargon” term. Having said that being able to determine new emerging terms relating to cyberattacks, malware, and other general information threats are useful if it can be automatically detected and incorporated into a threat system. In our own system, we outline a method to extract emerging attack terms from our data stream, but these require manual checking and investigation to confirm an appropriate term is added.

Hughes et al. [52] look at a method of identifying dual meaning terms using NLP. By comparing texts preceding and following an event, burst and dynamic topic models have been used to detect trending topics. The log-odds ratio indicates whether terms are more likely to appear in a given corpus, intuitively a new trending term will be much more popular in the target dataset compared to previous periods and other terms within the target period. For comparison, the log odds tools were compared against TFIDF with both methods including tokens related to specific cybersecurity events but with TFIDF also including unrelated terms. For this method to work effectively an appropriate prior and target time window must be manually selected which represents the before and after stage of the emerging term. This manual selection of the time period inhibits a total automatic solution which is sought after in an end to end system. In our proposed system we also use specific time windows to find emerging terms and measure their importance in the corpus using a combination of TF and TFIDF. For our testing period we have found that a one week period worked best for finding emerging topics in our dataset but this is an effect of the amount of threats which appeared during that time and could differ during more active periods.

1.5.2 Attack Sentiment

Sentiment analysis is a useful tool that allows us to place a polarity value on a document. When we consider this in the context of the cybersecurity domain we can use this information as an extra dimension to gain insights into possible attacks occurring. For instance, in the real world, a growing amount of hostility, unrest and general negativity can often lead to acts of violence and or even riots within the community [133]. Similarly, we can think that within the cybersecurity community a large amount of negative sentiment towards a company, government, or group which can be measured by the general social media posts towards this entity can end in some type of attack occurring towards them.

This can be seen in the group anonymous which has on many occasions announced attack campaigns against organizations that they deem to be causing some type of injustice or are in a morally wrong position.

In [15] they look at the correlation between sentiment and attacks conducted by website defacement hackers. This study saw that as may be intuitively expected there was a negative sentiment prior to the attack happening which conveyed a motivation for the attack. This was followed by a positive sentiment once the attack was conducted portraying a sense of relief or happiness. This pattern was found in 24% of the defacers and although this example applies to individual actors we can consider that negative sentiment within the greater security community will increase the likelihood that there is an attack on that organization.

In Hernandez et al. [49] they propose a methodology for tracking social media data which can cause cyber attacks, this is done through the classification of tweets based on content related to security attacks. The specific feature which is used in this study is the sentiment of tweets which are processes to find a correlation between the sentiment of tweets and their likelihood to be an indicator of a cyber attack. Based on this premise we can predict the occurrence of an attack based on the sentiment of tweets in a certain time window. Multiple machine learning models are tested to label tweets with a polarity value of negative, positive, and security-related. This method was evaluated using the 2016 USA presidential campaign where it was able to identify events where the twitter sentiment was affected and cause response from users. This shows that negative sentiment on social network platforms such as Twitter can be used as a warning mechanism for cyber threats. In our research, we also use sentiment analysis of tweets as an indicator of possible cyber threats, but in our case, we track tweets that contain certain attack keywords in correlation with organization names to pinpoint where attacks are more likely to hit. By having more specific data about possible targets we can more effectively defend ourselves and mitigate false-positives.

1.5.3 Ensemble Learning

Within this research one of the techniques we use to improve the classification and practical implementation of our method is by using machine learning ensembles [35]. By applying different machine learning models to subsets of our dataset it is possible to achieve superior results to that of using a one size fits all approach. By doing this we are able to apply more targeted solutions to each dataset based on its specific features. The benefits gained do not only help with classification score but also have practical implications in terms of speed, infrastructure, and scalability which affect how a CTI system can be implemented under different constraints.

There have been previous researches which have looked into the use of ensemble learning within the cybersecurity domain and how they can be leveraged for improved results. Zhou et al. [142] present an attack detection technique targeted at XSS attacks. This method proposes using an ensemble method where bayesian models for both domain knowledge and threat intelligence. In Miller et al. [87] they consider that an attack can be represented by various network characteristics which can be used as features for machine learning model. By creating ”perspectives” which are made up of features that support the same characteristics, there will be efficient diversity between these groupings. Various combinations of machine learning models were applied to these perspectives to gain an increase in classification scores. In Zhou et al. [143] they propose an IDS framework that takes advantage of specific features through an ensemble learning method. In this research, the optimal feature group is selected based on the correlation between features. They utilize an ensemble of classifier models such as C4.5, Random Forest, and Forest PA along with a voting method that makes a decision based on all of these classifiers.

In our research, we take a feature segmentation approach to enhance the level of our data. We use subsets of our dataset similar to the ”perspectives” mentioned based on common features such as length and platform, and apply different models to each subset to achieve an overall better classification score and speed.

1.5.4 Cybersecurity Datasets

Traditional features are comprised of things like IPs, port numbers, and other various network-related metrics. Originally, available datasets reflected this with the majority of the datasets traditionally used for the application of machine learning in cyber analytics being based on various technical datasets [136][130][90][23]. Because of the new and emerging nature of data mining open source text posts for the cybersecurity field and threat detection, more datasets have been released related to cybersecurity posts. Lim et al. [66] present MalwareTextDB, which uses a database of annotated APT reports to help construct models which can help research in data collection. Behzadan et al. [17] present a framework for detecting cyber threat indicators in Twitter data streams which uses a custom open-source dataset of 21000 annotated cyber-related tweets which is released to promote further research in the area. Pastrana et al. [96] present the CrimeBB dataset which contains more than 48 million posts from underground forums and stores it in a usable dataset that can be used to gain insights into the activities which lead actors into cybercrime. These are just some of the datasets which have emerged in recent years which have helped researchers study the cybersecurity field using NLP methods. Even though these datases exist, Zheng et al, [140] show that less than one-fifth of datasets present in cybersecurity research papers which are published are publicly shared. This has a negative effect of not allowing results to be validated and expanded upon leading to greater improvements and insights into methods. By making our data available we will able to improve the involvement within this research community and find comparison datasets that can become standard across multiple types of research and enrich existing studies and push them forward. To contribute to this we make our datasets available for public use.

1.6 Thesis Structure

This paper is made up of seven chapters that present the research we have conducted and go into detail into the methods used for each component. Firstly, in chapter 1 we have introduced our research topic and shown why it is important to pursue this research and the benefits that can be achieved through it. This chapter also presents useful background history which has led us to the current position we find ourselves in and which has influenced our decisions to conduct this research. Within chapter 1 we also present a comparison of our research with other related research to help situate where our work fits within the greater academic literature. By doing this we can explain the reasoning behind the choices which were taken for our research based on similar research.

In chapter 2 we present the common methods, tools, and tasks which were performed throughout our research. We introduce the broader concepts of various data science and data processing methods in this chapter so as to introduce the reader to the concepts which we use throughout our paper. When required we go into further detail as to the specifics of each method and its implementation in the appropriate section.

In chapter 3 we introduce our first research contribution by presenting our CTI framework. Based on the reasoning we have provided in chapter 1 we have developed a framework for a system that can leverage open-source data in the form of social media posts to proactively investigate and preempt cyber threats. This is achieved by using various methods and processes on the data which we retrieve and displaying it to the user so that they can easily take in and understand the output of our system.

In chapter 4 we cover our second contribution of this paper which is an extension of the data mining and filtering component of our CTI framework introduced in chapter 3. In the process of creating our system, we found that the data which is retrieved and used within this system is the most important component. Because of this, we develop our own method for filtering text-based cybersecurity data which expands upon the keyword filtering method. We create a method called Multi-Layer Keyword Filtering and Clustering (MLKF+C) which is able to classify documents based on relevance through a multi-layer keyword filter and then classify data based on significance through topic clustering. Using this method we can create a stream of high-quality data that can be used within our system and other systems to improve the usefulness of output.

In chapter 5 we present the third contribution of this paper which considers the implementation of our system in practical applications. It also considers the adoption of this type of system and how that can be affected by other factors such as business-specific aspects. In this chapter, we consider the practical application of our system and how it can be applied to the widest scope of scenarios possible. We do this by investigating the use of ensemble learning on our data and how this can promote a more favorable trade-off between classification score and processing time. We also look at factors such as scalability, maintenance, infrastructure, budget and show how using ensemble learning we can create various architectures of our system to encourage wider use of our system in organizations that previously would not have had access to this technology.

In chapter 6 we discuss the various components of our research and how these components can be configured to suit various scenarios. As part of this, we also consider the scenarios where our methods perform sub optimally and what are the factors that cause this. Finally, chapter 7 presents the conclusion of this paper and summarizes our work as a whole.

Chapter 2

Methodologies & Tools

In this section, we present the common techniques and technologies used in our research. Since there are certain components that are shared across our research, rather than present these aspects across multiple sections we will introduce them in this section and then go into any further detail if needed in the respective section.

2.1 Data mining

With the rise in internet-connected devices and the prevalence of technology has come the inevitable rise in data. This data is not only restricted to photos, videos, and messages on social media but includes a number of protocols and control data which allows for internet-connected services to function. The amount of data flowing through our networks in a single day dwarfs all previously recorded physical data in the form of books. Digital data is much more robust compared to physical data, since it is able to be copied to databases across the world in seconds. This has brought with it the added benefit of allowing us to analyze this information as a whole, which has come to be known as data mining. Data mining refers to the analysis of big data, usually in the form of large datasets and the subsequent process of finding relationships and patterns within that data which can provide some insight into the underlying nature of the data which is useful in some way.

This process provides specific challenges such as how we store, access, and read data. Due to the sheer amount of processing speed required, it must be considered to determine if it is even possible to deal with the whole dataset or if a sample will be used. Also from an analysis perspective due to the large amount of data that is used there will be a larger amount of edge cases that find their way into the dataset. Therefore identifying these cases and accurately reflecting and summarizing the true overall trend of the data is important.

Data mining is the analysis step of knowledge discovery in databases (KDD). KDD is comprised of various steps which help outline the process of finding and interpreting relationships from data. These steps include the selection of data that is applicable to a goal, pre-processing the data to fit requirements and allow for processing, data transformation to transform the data into a specific type, and, performing data mining to extract the relationships and patterns present in the data. Finally, these patterns and relationships can be analyzed and interpreted to gain insights into the data as a whole. Within our systems, we use these established steps as a way to extract useful insights from our data.

2.2 Dataset Creation

The first step in this process is the acquisition of data in our particular domain. Although as we have previously stated there is an abundance of data currently being used throughout the internet, accessing this data is not always an easy task. Companies go through a large amount of work and employ teams of people to capture, store, and maintain this data, and it is not always in their interests to provide open access to it. The insights that can be gained from these large datasets are incredibly valuable and can easily supply a company with insights that give it an edge against its competitors. Although this is common and many closed repositories exist which do not provide access to their databases to outside users, there are cases where companies provide access to a subset of their data. This is often done through an application programming interface (API). This programmable interface allows outside users to interact with a database or other underlying infrastructure using specific calls or requests which the developer has allowed. In this way, a user can gain access to a large database that is maintained by an organization and use it as a data source.

One example of this is the Twitter API [75]. The Twitter API allows users to access data about specific tweets, users, messages, trends, and more. This is done through the API interface which allows Twitter to set what parts of its platform can be accessed and in what way. This access is provided to users because it expands the adoption and general use of this platform. Through this API outside developers can create applications that integrate Twitter data into them. Research can be conducted using this data to gain new insights into specific areas and many other applications can also be found. This is beneficial to the provider since it expands the reach of their platform and integrates it into existing systems ensuring its use and longevity.

In our research, we take advantage of the Twitter platform using the Tweepy library [106] to create a dataset of cybersecurity-related tweets. We are able to use this interface to retrieve a stream of tweets based on the existence of cybersecurity terms contained in either the body of the tweet or as a hashtag. That is to say, we can filter and retrieve specific tweets that contain a cybersecurity-related term in them and subsequently save that to a database to create a repository of cybersecurity-related tweets which we can further process and analyze. We use this technique throughout our research to collect data. Twitter is a platform used throughout the world which allows us access to data that can give us good insights into threats.

Although Twitter is a good platform to access cybersecurity data it is not the only source that contains relevant data that can assist us in getting insights into cyber threats. There are various inherent aspects of Twitter that will exclude certain types of data from appearing on that platform. For example, Twitter posts are restricted in length to 280 characters. This shorter length generally lends itself to more grammatically unstructured text with more emojis and deliberately misspelled words. Also, although Twitter is used throughout the world, Twitter like any platform does not represent the only option that can provide us with relevant data.

It is important for CTI systems to be able to efficiently process and gain insights from different types of data. Because of this, it is important to represent other text styles in our dataset to ensure that we cover the widest scope possible with our system and are not limited to any particular platform or type. To do this we also integrated other data into our MLKF testing procedure. We integrated data from four unique platforms to create this dataset using Twitter[1], Reddit[2], Stack Exchange[3], and traditional news articles[4]. We chose these specific data types because each has its own unique characteristics in length, style, and content which give us a good overview of styles, and allow for our results to be applicable to other platforms.

Stack exchange is a popular network of question and answer websites on various topics, with each subsite being dedicated to discussion on a specific topic. There is an open-source tool called Stack Exchange Data Explorer [40] which provides access to public data from the Stack Exchange network through SQL queries. We utilized this tool to retrieve data that was relevant to our use case based on the specific domain of the site in the network, and also tags that were assigned by the user to particular questions. In this way, we were able to retrieve relevant cybersecurity data for our research from Stack Exchange. Stack Exchange data is unique within our dataset because of its particular qualities. The questions posted through Stack Exchange are often technical in nature containing large amounts of programming code or markup language.

Reddit is a network of community discussion websites based on common interests. It also has specific subsites that are dedicated to particular topics where users can discuss events within that domain. Since particular subreddit discussions are moderated to be

[1] https://twitter.com/

[2] https://www.reddit.com/

[3] https://stackexchange.com/

[4] https://www.welivesecurity.com/

relevant to the theme it is dedicated to, we can use this as a filter for our data. We use specific cybersecurity-related subreddits such as r/netsec and r/cybersecurity to retrieve data that is relevant to our applications. We are able to access this data through an open-source tool which stores historical Reddit threads called pushshift.io [16]. Reddit data is made up of a range of styles from general to technical discussions based on the subreddit.

We also retrieve traditional news in the form of news articles from cybersecurity-related news websites such as welivesecurity and the-hacker-news [39][91]. We retrieved this data using the Python library selenium [1]. This provides a different text style compared to the other entries with news articles being longer and more structured compared to the other entries.

This dataset is aimed to be used within the realm of ML and the training of ML models. Because of this both positive (cybersecurity-related texts) and negative (general texts) must be used to train the model. Therefore to retrieve negative examples for our dataset we used the same techniques as outlined for each platform but use those techniques to retrieve data which was general and did not contain any reference to cybersecurity events. In this way, we were able to create an automatic cybersecurity-related dataset based on open-source data platforms that were related to cybersecurity posts, questions, and discussion data.

2.3 Data Cleaning & Pre-processing

Data cleaning and pre-processing is an important step within the KDD pipeline. It can employ various techniques to normalize data into a form that is able to be effectively used for analysis. These processes can differ depending on the ultimate goal to be achieved and the type of data that is being used. In our case, we are using text data in the form of text-based posts and articles. Therefore our cleaning and pre-processing techniques aim to get our data in a form that will assist the models we use to more easily be able to identify the key concepts in the text.

The first step in this process can be implemented during the dataset creation phase while retrieving data or afterward by processing the data once it has been stored. This step includes filtering out entries which are either missing data such as empty fields, fields which only contain out of range data, or fields which have impossible data combinations. In our case, our data is made up of four fields. An entry contains the timestamp for when the particular post was published, the text contained in the post itself, the platform from which the data came from, and a field that captures a sentiment score.

Based on these fields if we have data that has a missing body field there would be no reason to add this entry into our dataset as doing so would only diminish the quality of the dataset and lead to questionable results. There are cases where entries with missing values will still be used but the missing value will be automatically replaced by some placeholder value. This technique can be used for small datasets where data is scarce. In our case we do not have this issue and therefore can filter out data entries that are missing values or have any similar issues. Once this has been done we have a dataset that contains a large number of entries from which we aim to analyze and extract patterns and relationships. Because we are working with text data there are further processes that must be taken to mold the data into a form that is more suited towards being used in a classification model.

The initial steps which are commonly employed in the pre-processing stage for text data and which we employ begin with vocabulary minimization. We first remove stopwords from each entry in our dataset. Stopwords refer to common words like, is, at, or, which, etc. These terms can cause issues in phrase searching and are not beneficial for the method which we are employing. By removing stopwords from the body field of our data we are able to also minimize the vocabulary size which assists in processing time.

The vocabulary of our dataset refers to each unique term that makes up all entries from the body field of our data. A large vocabulary can cause longer processing times and can cause various sparsity issues. We therefore want to minimize our vocabulary to a point where it is still able to efficiently describe the underlying concepts contained in the text but it does not contain a large number of non-relevant entries. This issue can vary in significance depending on the platform. Although news articles are generally longer than other types of online posts they are written using structured and formal language and have been edited and checked for misspelling. On the other hand, the unstructured format and unusual spelling used on a platform such as Twitter means that it has a much larger vocabulary size due to all the iterations a word can be spelled as and the countless combinations of characters that are used.

Because of this, it is important to further pre-process these platforms to remove unintelligible strings of characters and other platform-specific elements which do not hold value towards our final goal. This includes elements such as @mentions, hyperlinks end emojis. Although there has been research done into how some of these elements like emojis can be utilized to assist with areas such as sentiment analysis we do not use these features in our method.

Another factor that contributes to greater vocabulary size and has an impact on classification is a large number of various forms of the same word. There are derivations of the same word such as works, worked, working which can be reduced into one representative form. This procedure can help by reducing the vocabulary size and helping to search our data based on a common base term rather than having a large range of terms with the same meaning.

There are two main techniques that can be used to reduce terms to a common base form, stemming [70], and lemmatization [100]. Although both have similar goals they both achieve them in different ways. Stemming is a simpler technique that cuts off the end of the word without any consideration for the meaning of the word itself. Because of this it can be somewhat crude and can leave strange terms in the vocabulary. Although, because it does not consider the meaning of the terms itself it is quicker to apply and therefore can be applied to larger datasets with less processing time. Lemmatization is similar in that its aim is to reduce the words down to a base form. Lemmatization in contrast does consider the meaning of the word by using a dictionary and aiming to remove inflectional endings only, leaving the dictionary form of the word behind. Since this method does take into account the meaning of the word and utilizes a lookup dictionary on each word it can take longer to process a larger dataset than stemming, but can return more consistent results. For our separate researches, we test the effect of stemming, lemmatization, and original data on our separate datasets. We go into more detail on how these techniques affected classification separately in their respective sections.

Up to this point, we have removed various elements from our dataset and tried to reduce the size of it to condense it into an expressive form that efficiently represents the underlying concepts of the data without any unnecessary entries. This is all done with the aim of assisting and improving the classification of our data. We can also improve the classification of our data by manipulating the existing data in our dataset to create new features. One common technique to do this is by creating n-grams. N-grams refer to a sequence of terms of length n. By using n-grams in our filter we can improve the quality of our system by reducing false-positives and using a combination of n-grams allows us to get the best results from our method.

Using these various data cleaning and pre-processing methods on our dataset we can ensure we have useful data that produces reliable and consistent results. Without using these techniques it is likely that the system would be strongly affected by the quality of the data which was input into it and therefore would could not be relied upon to give useful or meaningful output.

2.4 Data Transform

In the previous step, we explained how we can clean and pre-process our data to remove noise, outliers, and missing data. We also show how we improve the existing data and expand it to better represent the underlying relationships within. Once we have converted our data to get it into an effective and efficient representation, we need to transform the data into a format that is able to be used by the machine learning models and other methods we are using.

It is often the case that many machine learning models can not take raw text-based data in its original form, instead, they expect a fixed-size vector of numbers. This means that we are required to transform that data into a collection of tokens that accurately reflects the dataset. This process includes the tokenization of strings by assigning a numerical value to each token, the counting of these tokens, and the normalization of tokens based on the count. The vectorization process allows us to transform our dataset into a form which we can then input into a machine learning model.

For our purposes of text classification, we test two techniques to transform our dataset into an appropriate format. Firstly, we use the Scikit learn [97] function CountVectorizer. CountVectorizer implements tokenization and occurrence counting in one class. Each unique term in the vocabulary is assigned an integer representation and entered into a matrix that represents the particular document. This technique is usable but does not consider term frequency and importance which will end up having an impact on classification. Certain terms will be found more often in a corpus, terms such as the, a, or, will far outweigh more descriptive terms which hold more meaning within the corpus. The over-representation of terms that have little significance within our data will obscure our results and stop us from getting clearer outcomes. To fix this we can weight our tokens based on a value that represents the importance of the term in a corpus. The TfidfVectorizer function performs the same process as CountVectorizer but instead of assigning a count value only based on occurrence, it applies a value based on importance called TFIDF [101]. This is achieved by multiplying the frequency of a term by the number of documents that do not contain that term.

$$\text{tf-idf}_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t} \tag{2.1}$$

This in turn should create a better representation of our data since it is including the aspect of term importance in its transformation rather than simply distinguishing between different terms. We test these two techniques in our research to see how they differ and what effect they have on our data in terms of classification score.

These techniques provide us with a vector of features that can be very long. This can become an issue if it is too large, meaning further transformation can be beneficial and even necessary for certain tasks. There are various dimensionality reduction techniques that take a high dimension vector and look for a combination of features that best represent the original features within a smaller vector size. By doing this we can reduce the size of the matrix representation of our corpus into a representation that can be used with particular models. For example, there are certain algorithms that are only able to handle vectors of a certain length, so if there is a vector longer than what is accepted by the algorithm, it can be reduced with this technique. Also, to visualize our data to gain better insights we may plot it in two or three dimensions and therefore have to use an implementation of this technique such as Principal Component Analysis (PCA) [135].

In our case, we test various vector sizes in our research to see how the various representations affect our classification and at what size we can retain enough detail of the underlying relationship of the data while still saving space by using a smaller length. We also visualize our data to gain further insights into the similarities between various data entries and therefore have to use these techniques for data transformation.

2.5 Data Mining Tasks

Depending on the goal of the task different processes will be used such as classification, clustering, or regression. Within our research, we use various tasks within our system to obtain our desired output. We explain which tasks we use and their purpose in this section.

2.5.1 Sentiment Analysis

Sentiment analysis refers to the analysis of a piece of information to identify and quantify the underlying emotion behind a particular piece of data. This technique can be applied to various types of data such as voice recordings and text. In our particular case, we are dealing with various types of text-based data and therefore use this technique to classify posts with a sentiment value. The typical use case for sentiment analysis is to assign a sentiment score between 1 and -1 to a piece of data. This correlates to an emotion such as positive, negative, or neutral which reflects the emotion of the data. This task classifies the polarity of the text and can be expanded to identify specific emotions such as happy, sad, anxious, and others.

Our framework uses a sentiment classification model to assign a polarity score to our tweets giving us an extra dimension that we can use to analyze our data. By identifying data points that are classified as having a negative sentiment and that also contain a particular threat name such as "DDoS" we are more likely to encounter posts that relate to attacks. By tracking posts that are classified with a negative polarity and contain specific attack names we can get an overview of the threat landscape for that particular type of attack. If we take this further we can also identify specific company names in this stream to help us identify posts that refer to specific attacks on companies.

By tracking posts that meet this criteria we are able to get a indicator of what threats are occurring on different companies in real-time. This information can then be used by an analyst to assess the risk in their own network and adapt to real-time attacks. We implement this technique in our framework for proactive cyber defense and discuss the results obtained in Section 3.1.3.

2.5.2 Word Embedding

One of the areas which has benefited from the increased research into ML and artificial intelligence has been NLP. NLP is concerned with the problem of allowing computers to process and analyze large amounts of language data. One way to do this can be using a language model to distinguish between words by using a probability distribution. An example of a type of language model which does this is a word embedding model. Word embedding is a technique where an individual word or document can be represented by a vector. By representing individual words as vectors we can therefore measure the distance between them and correlate this with how similar two words are. It also allows us to perform algebraic functions on terms such as King – Man + Woman = Queen.

This is achieved by looking at the context of a word, meaning, the words it is commonly used in conjunction with. By looking at a large number of examples of words that are used in conjunction with a particular word we can find the probability of which terms are most common and then represent this in a vector form. Other vectors which have similar values mean they have common terms and thus share a similar meaning. The opposite is also true with vectors that have no shared values or are far apart from each other having little similarity. Word2vec is a particular implementation of this method first presented by Mikolov et al, [86] and is the implementation we use in our research. We use the word2vec model to disambiguate terms within our corpus. By using a trained word2vec model we generate a list of terms that are similar to a particular term from our corpus. We can then use this list to distinguish if a word is relevant to our domain. In this way, we can create a filter method to include entries that are relevant and discard data that is not. The word2vec model has various hyperparameters which can be adjusted to increase performance and classification. It is also affected by the quality of inputted data through pre-processing and transformation. We look at all these aspects and test how each component affected our overall classification score.

2.5.3 Clustering

Clustering is the task of grouping various data points together that have common features compared to data points in other clusters. It is a common technique used for data analysis and classification and encompasses various techniques that implement various solutions

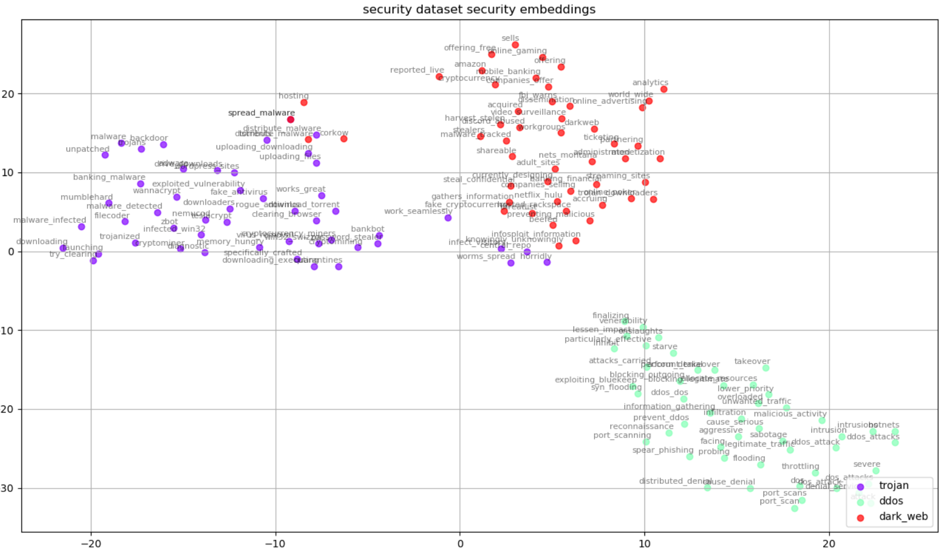


Figure 2.1: This figure shows the top 50 terms by cosine distance for the terms, trojan, DDoS and dark_web. By training our model on cybersecurity-related data we can use our word embeddings to determine the context of posts.

to achieve clustering. Within our research, we use clustering on our dataset to identify groups of data that contain similar topics. By identifying groups of data that are related to the same subject matter, we can also distinguish non-relevant subjects within our domain and attempt to extract those clusters to improve the quality of our data.

To achieve this there are various clustering algorithms which we test throughout our research to find which is able to best distinguish the underlying topic of our data. We test algorithms such as spectral clustering [92], K-means clustering [48], and fuzzy cmeans [20] to evaluate how each algorithm interacts with the data itself and if they can be tuned to provide better results.

A challenge with K-means and other algorithms is the need to provide the number of expected clusters to the algorithm. Our dataset is not labeled by theme and therefore we do not know the number of clusters we should assign to the K value. Because of this, we turn to metrics that do not require a ground truth table. In the case of clustering, cluster separation and similarity analysis can be used in the absence of labeled data. Silhouette score and Davies-Bouldin index [98] are two metrics which we can use to evaluate our clusters without having labeled data to compare against. Silhouette score and Davies-Bouldin index provide us with the number of clusters but the final judgment for our particular application comes from finding which clusters contain the type of data we consider to be less significant. To identify this we can analyze our clusters to see which terms are most prominent within each cluster, we do this by ranking terms in each cluster based on TFIDF score. This allows us to see an overview of the themes and types of posts in each cluster. For example, if a particular cluster contains a large amount of terms such as news, latest, via or similar terms common in advertisements we can obtain a higher quality dataset by removing that cluster from the dataset.

2.6 Performance Metrics

To evaluate our methods we require a method to quantify our model that reflects the model 's goals. From the confusion matrix table, we can calculate various metrics, the most commonly used metrics are, accuracy, f1-Score, recall, and precision [61]. If we look at the CTI systems previously shown [64][108][83][144][93], these systems use one or more of these metrics to evaluate the performance of their model. In some cases [63][88] they use a more holistic approach by manually analyzing the events created by the model and evaluating their quality. Using the correct metric for a particular dataset is important to achieve quality output, the metric chosen should correlate with the metrics score and the quality output of the system if it is chosen well.

When choosing which metric to use we need to consider what type of performance we require from our model. To evaluate a model 's quality we can use any of the measurements previously mentioned, each measurement will lend itself differently to our application making it important to choose an appropriate metric to not be misled by the score. The accuracy score represents, "the probability that a classified post is correct". Precision represents "the probability that posts classified as relevant are actually relevant" signifying how often we correctly classify posts. The recall represents ”of all posts classified as relevant, how many were detected”, and finally, F1-score is the harmonic mean of the precision and recall, giving a trade-off between the two metrics.

1. Accuracy = The probability that a classified post is correct.

2. Precision = The probability that posts classified as cybersecurity-related are related.

3. Recall = Out of all posts classified as relevant, how many were detected.

4. F1-Score = F1-score is the harmonic mean of the precision and recall, giving a trade-off between the two metrics.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives} \tag{2.2}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.3}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.4}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

We need to identify what is the most important metric for our system, this could be to filter out all non-relevant posts at the expense of legitimate posts, or make sure all legitimate posts get through at the expense of also letting non-relevant posts through. For example, if we apply the metric of recall and assume, from 100 posts, 80 are relevant and 20 are not, classifying all 100 posts as relevant will give 100% recall score even though 20% of the data was miss-classified. This shows recall on its own is not a good measurement for this system. The accuracy measurement is often used in research but can be misleading. Depending on the platform it may receive varying amounts of relevant and non-relevant data, giving an unbalanced distribution. If a platform receives 98% relevant and 2% non-relevant data in a day, the accuracy is 98% which is misleading if our aim is removing all non-relevant data.

This shows that choosing the correct measure to evaluate performance is crucial. Choosing the wrong metric can cause misleading performance scores and ultimately a defective system. Because of this we must identify the particular use case of our model and select a metric that caters to that use case. Because of the amount of data being input into the system we want to maximize the number of correct classifications using metrics such as f1-Score and precision. By maximizing true-positives we can ensure the majority of our data is relevant, by achieving a high score, any misclassified data will be "drowned" out by the amount of correct data which will show the greater trend. Although we are trying to utilize this method to assist in false-positives, ignoring ambiguous data means severely limiting the scope and ultimately lowering the usefulness of the system. Because of the amount of data being input into the system we want to maximize the number of correct classifications using metrics such as F1-Score and precision. F1-Score is a good indicator of performance in our case because it takes both precision and recall into account. Depending on the application, precision alone can represent performance well but in our case, it has the effect of maximizing true-positives by not processing false-positives which leads to the exclusion of ambiguous data which is not beneficial in our scenario. Therefore we focus on f1-Score as the main metric throughout our research but also consider the insights that can be gained from precision.

Chapter 3

Cyber Threat Intelligence Framework

In this section, we present a framework for a data-driven threat intelligence system that is designed with the aim of increasing the capabilities of its users to become more proactive in its cyber defense. We believe that the increasing amount of data which has been generated by the emergence of new platforms and services on the internet can drive a more proactive stance towards cyber defense which will allow us to move away from the traditional reactionary stance.

3.1 Method

In our design, we include various components that need to be considered when implementing an effective CTI system. We present a framework for a system that is able to filter data from social network platforms and take advantage of the greater context which is inherently contained in these data points. The data filtering process takes various aspects into consideration to implement a method to effectively filter data. It also considers factors such as the emergence of new attack names and how these terms can be integrated into our existing filters. Once we have filtered and obtained our data, processing and data

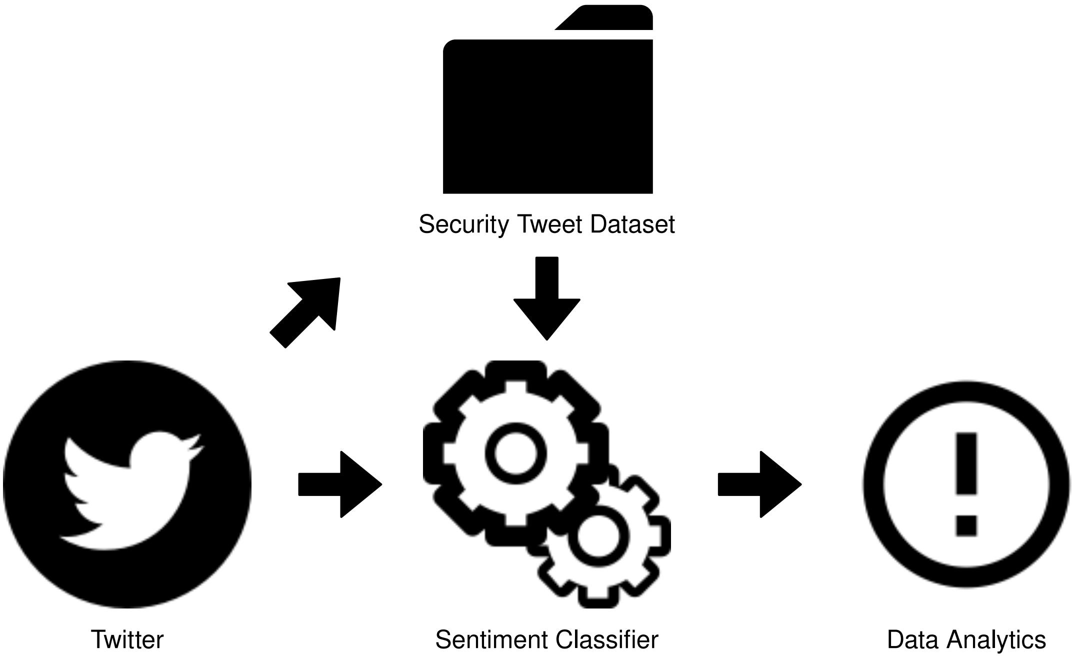


Figure 3.1: This figure shows a simplified version of the principal components for our proposed framework.

mining is conducted. This is done to allow us to find patterns and relationships within the data which would otherwise not be found if we were manually analyzing the data. Finally, we look at how we can effectively and intuitively convey our results to an analyst in a way which they can quickly understand and act upon.

3.1.1 Data Filtering

This component of our system aims to filter out cybersecurity data from a greater data stream of Twitter posts and store those data points for later use. Our first action to implement this is to consider the legitimacy of the data. In recent years there have been many questions as to the authenticity of certain social media posts and whether they can be considered legitimate or ”fake” [10]. By not integrating ”fake news” into our system we can avoid erroneous output that does not accurately reflect real-world events. To do this we can take advantage of features that come inherently in Twitter data. Twitter post data contains various features that are associated with it such as the user who posted the data, the time it was posted, the amount of retweets, and more. Therefore to ensure the legitimacy of our information we begin by filtering our data stream based on accounts related to well-known researchers, hacktivists, companies, and other prominent accounts within the cybersecurity community. We can expect that the posts which come from these accounts are legitimate due to greater public scrutiny on these accounts because of their prominent placement.

At this stage, we have a stream of posts which comes from various cybersecurity-related accounts that we consider to be legitimate and usable. Although this data stream will contain a large amount of related data it has been shown that researchers and enthusiasts will likely post about topics that are not security related [64]. This can include topics such as conferences and general life events. Therefore to further filter this data stream and limit it to security-related topics we use a keyword filter that contains relevant terms that can be used for monitoring social media platforms. Al-Rowaily et al. [8] use the ”2011 analysts desktop binder” which outlines common security terms that can be used in this scenario to monitor social media discussion. Due to the age of the list, we use it as a basis and add new terms and attack names to it to bring it up to date.

Within this data stream, it is still possible to retrieve erroneous data. This generally takes the form of false-positives which are incorporated due to terms that have dual meanings. These terms are common in the cybersecurity domain such as backdoor, virus, and worm. These terms can be used in a context that does not relate to security and therefore when they are used in this context we do not want the data to be stored. To achieve this we use a separate static list of words that are commonly used in general cybersecurity discussions. By processing our dataset using TF we were able to obtain a list of terms that are commonly used in cybersecurity posts but which are not within our main list. From these terms, we created a separate list of words that we can use to compare against the body of the tweet. If the body of the tweet contains a word such as backdoor which can be considered ambiguous we then scan the rest of the body and

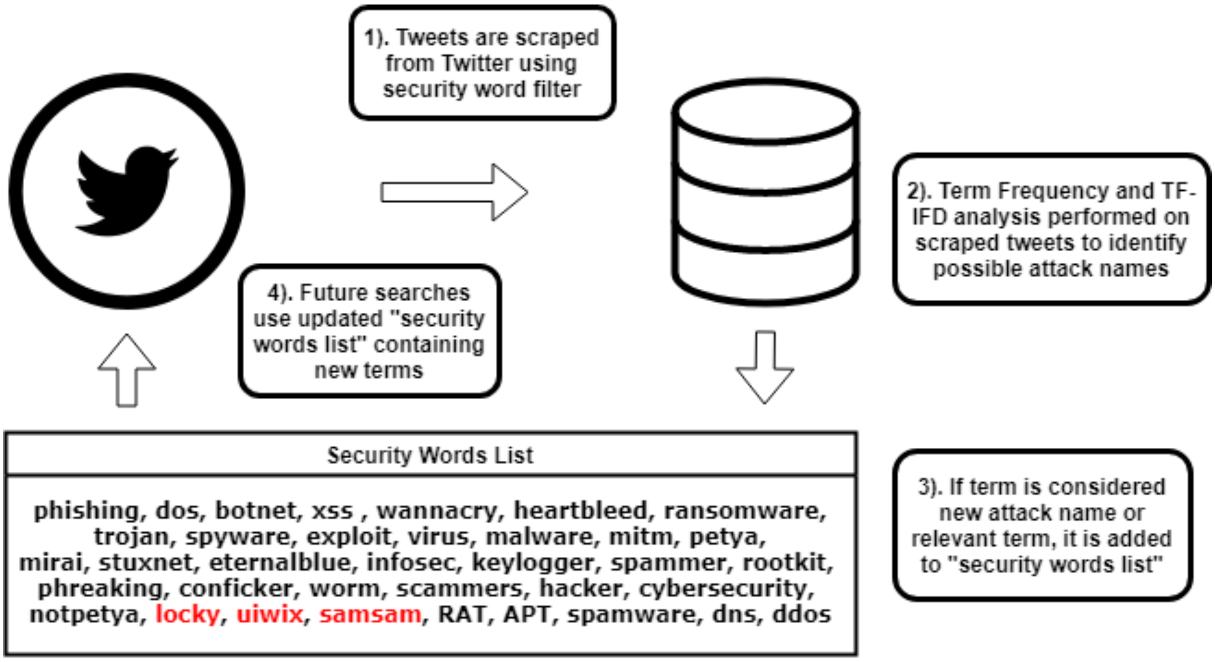

1). Tweets are scraped from Twitter using security word filter
2). Term Frequency and TF-IFD analysis performed on scraped tweets to identify possible attack names
3). If term is considered new attack name or relevant term, it is added to "security words list"
4). Future searches use updated "security words list" containing new terms
Security Words List
phishing, dos, botnet, xss , wannacry, heartbleed, ransomware, trojan, spyware, exploit, virus, malware, mitm, petya, mirai, stuxnet, eternalblue, infosec, keylogger, spammer, rootkit, phreaking, conficker, worm, scammers, hacker, cybersecurity, notpetya, locky, uiwix, samsam, RAT, APT, spamware, dns, ddos

Figure 3.2: This figure shows the method developed and used within our system to detect new and emerging attack terms which can then be integrated into our keyword filters

compare it with our second list to judge its relevance and either store or discard it based on this factor.

3.1.2 Emerging term Incorporation

As part of our framework, we consider how to deal with new and emerging attack terms in our system. Due to the fast-paced and highly evolving nature of the cybersecurity industry, new threats and thus new attack names are constantly arising. When using a method which filters data based on keyword filters these new attack names need to be added into the filter so as to keep the data that is being processed up to date and the output of the system current. In our proposal, we consider a lightweight solution that uses TF and TFIDF techniques to process our dataset to find new and emerging threat names.

When using this technique we have to consider the time period being used. The TF and TFIDF methods can be used as a way to identify significant terms within a dataset. If the scope of time from which the data is being taken is too large there may be an issue with various important terms being present within the time period and therefore making them more difficult to identify. Because of this, we found that by conducting this process on a dataset in regular intervals such as weekly or bimonthly we are able to more easily track new and emerging terms in the dataset. These terms can then be analyzed manually and incorporated into the filter to keep the system data up to date.

3.1.3 Sentiment Classification

Once we have ensured we have a legitimate and up to date stream of data we can begin to process this data to gain insights into it. This framework outlines a system that can be used to help security analysts be more proactive in defending their network and to do this we must be able to give a real-time view of the threat landscape at a specific time. Having information about the polarity of the tweets we have mined allows us to do this by giving a better contextual view of public opinion towards a specific entity in relation to a specific attack type. In Hernandez et al. [49] they find a correlation between posts which contain a negative sentiment on social media and cyber-attacks. Similarly in Aslan et al. [15] they look at the correlation between sentiment and attacks conducted by website defacement hackers. They found that a defacement was often preceded by negative sentiment, showing that these posts can be taken as an indicator of an attack.

With this in mind, we use sentiment analysis to classify our tweets with a polarity score between -1 and 1, allowing us to judge how negative or positive each document is. By analyzing the tweets we have based on negative sentiment and grouping them based on common attack terms we can gain a new metric into the likelihood of an attack occurring. For instance, if we have a large number of negative tweets containing the term ”DDoS” this can be an indicator of a ”DDoS” attack occurring. Furthermore, if we can correlate these tweets with a specific entity such as a company or government we can further pinpoint a possible target for the attack. An analyst can use this information to help prepare, inform and protect themselves for possible attacks before they happen and contribute to a more proactive stance rather than maintaining a reactive stance.

3.1.4 Visualization

One of the most important aspects of this type of system is the way it conveys its knowledge to the user. If we are not able to effectively communicate the outcomes of our system effectively it does not matter how accurate or insightful the output is since the user will not be able to grasp it well enough to implement the required changes into their network. To achieve this we consider the design principles which factor in too this. In [81], design principals such as the avoidance of 3D graphics and complex visualizations are recommended in security systems to help with ease of understanding of the visualization. In [41] they also mention the applicability of the visualization to large screens which are commonly used in operation centers and other scenarios where this type of information is required. We have created our visualizations with these principles in mind to help develop a usable interface that effectively conveys the insights of our data. In our framework, we consider the best way to convey the key outputs of our data to the user in a way that can be quickly read, understood and actionable. To achieve this we implement an interface that visually depicts the data we have obtained. By doing this we can represent large amounts of data that are taken in by our system allowing the user to gain insights which are not possible manually.

In Figure 3.3 we can see an example of the main interface which can be used to track multiple companies based on the polarity of their data. This allows a user to quickly understand what the current situation is in regards to a particular set of companies with the option to dig further into the data to find detailed explanations. This interface is intentionally simplified down in a way that allows a user to quickly understand the current situation. In a setting such as a security operations center which can be a large room with multiple screens in it. It can be difficult for the principal analyst to grasp detailed information quickly from a screen. It is therefore better to have easy to read and understand visualization which work at various levels to quickly grasp the current situation and make a decision based on that data [21].

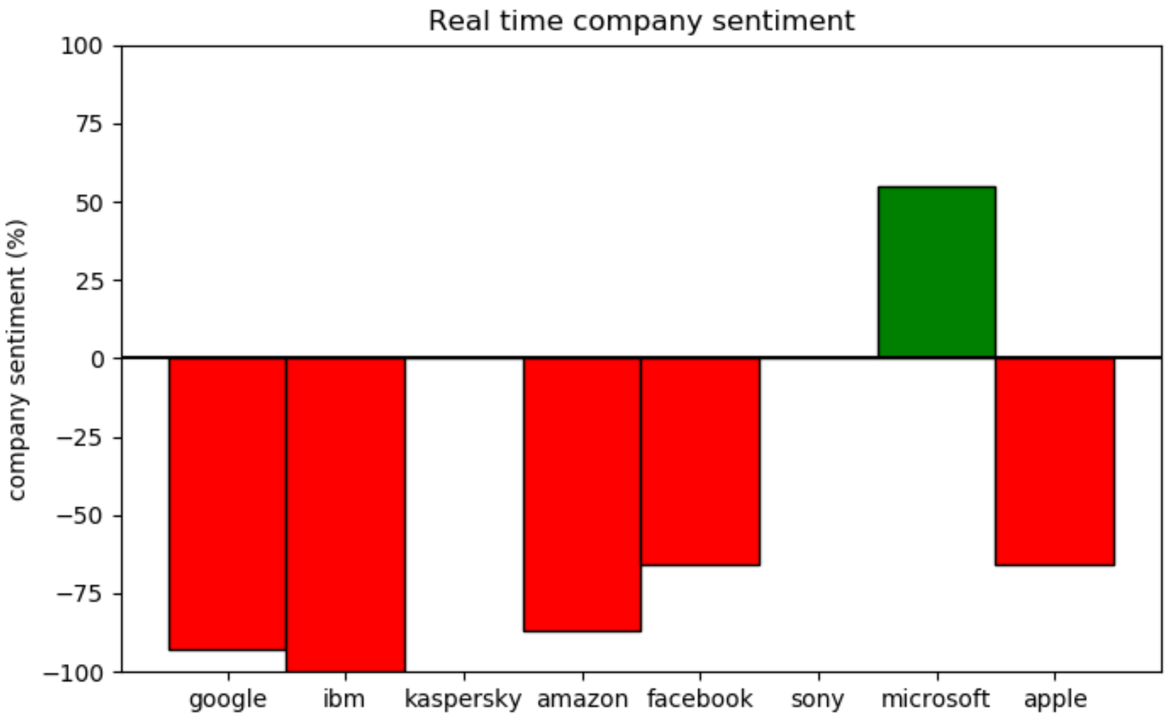


Figure 3.3: This figure shows the interactive user interface for our system which measures the average sentiment scores of tracked companies based on our collected data

Even though it is important to be able to quickly understand our data this does not discount the detailed aspects required which can be useful to a system administrator or technical analyst. Because of this although the main interface is made to be extremely clear we also allow the user to easily dig further into detailed results of our processes to gain better insights.

Algorithm 1 Psuedocode showing the algorithm used to filter normal tweets to security related tweets.

```
Input ← Tweets from security user accounts list
Output ← English tweets related to cybersecurity

if If tweet contains word in security_user_account then
    if If word is in double_meaning_words_list then
        if if tweet contains word in security_related_words_list then
            Storetweet
        else
            drop tweet
    else
        store tweet
else
    drop tweet
```

3.2 Results

3.2.1 Data Filtering

Our filtering method utilizes, user-level account filtering to obtain legitimate data. We then filter this data based on keywords to determine post relevance. Finally, we use a second filter based on the most frequently used terms in the dataset to judge if ambiguous posts are relevant or not. Once we put our dataset through this process we are left with 70,475 tweets which were considered usable from a total dataset of 1,716,787 general tweets that were retrieved. The algorithm used to filter for relevant data can be seen in 1. To show that the dataset we have created using this method is able to create a dataset that contains cybersecurity-related data we look at the principal terms in the dataset. In table 3.1 we can see the top 15 terms in the dataset by frequency. This shows that the most frequent terms within the dataset are related to security topics and show the relevance of our dataset as a whole. The terms which are not related and are seen in the top 15 list are considered stop words and are common throughout all discussions. By using text pre-processing techniques these terms can be removed during the data pre-processing phase. When creating our list of words that are used to find associated words within the body of the tweet we remove stop words and the most frequent and common attack names which give us a list of commonly associated words.

To further show the validity of this dataset we show a co-occurrence graph for the most common terms in our dataset which can be seen in figure 3.4. This graph shows the terms which are most likely to occur together in our corpus. Considering that all valuable relations occur in conjunction with a security term we can consider this to show that the relevant patterns in our dataset revolve around cybersecurity topics.

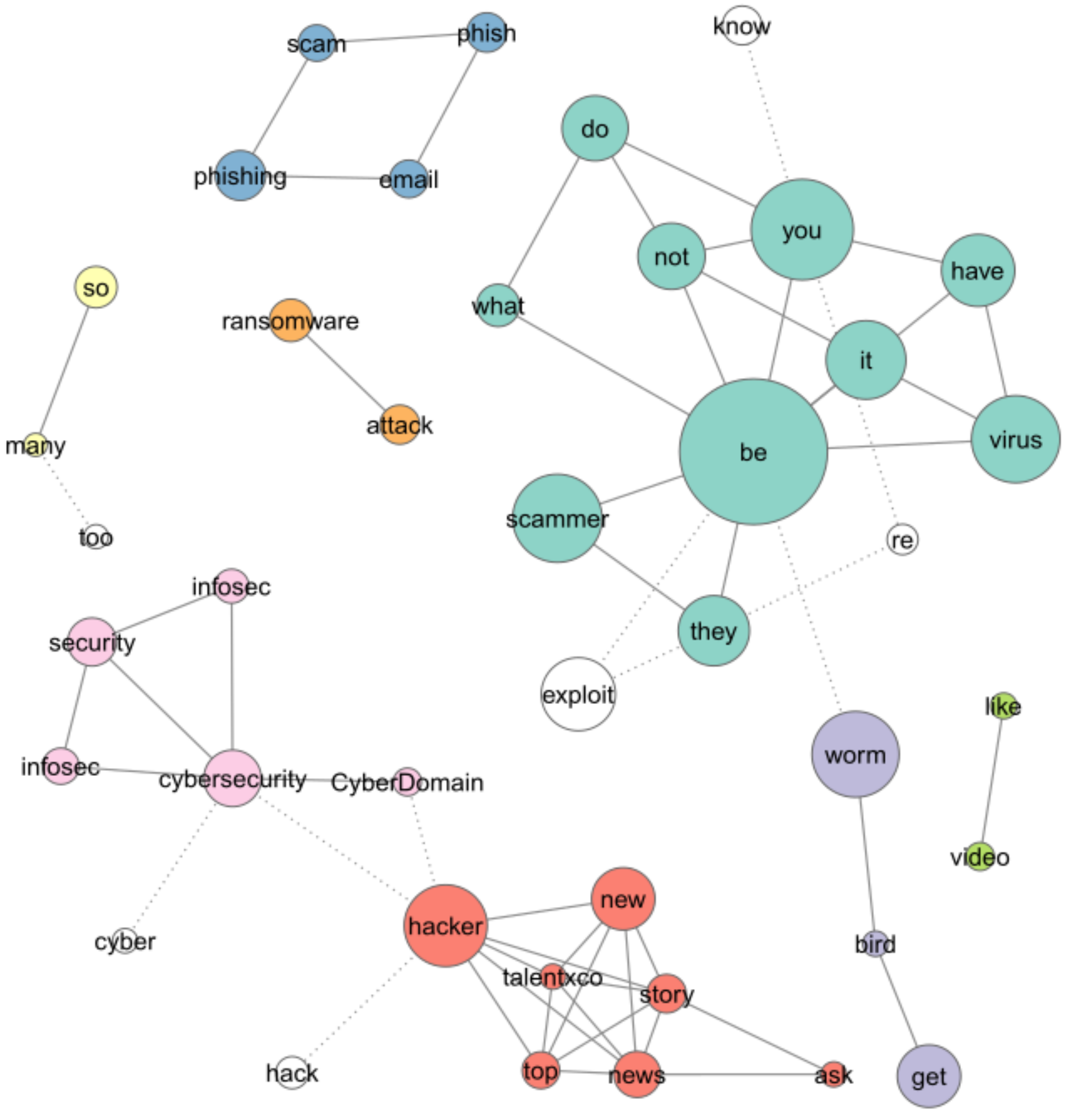


Figure 3.4: This figure shows the co-occurence networks for key terms in our created dataset depicting the relevance of security data within the dataset

3.2.2 Emerging term Incorporation

In the testing phase of our system, we performed TF and TFIDF processes on our dataset of 70,475 tweets in subsets based on weekly intervals. The period of time from which we take our data for this process has an impact on the discovery of emerging terms. If we perform this process on a short time frame such as daily, there is a higher likelihood of

Table 3.1: This table shows the top 15 terms in our created dataset by word frequency. Eleven of the top 15 terms are security related and therefore show the prominence of security posts within the dataset

Word	**Frequency**
scammers	13,181
virus	12,900
worm	12,468
hacker	10,522
exploit	8962
malware	7393
phishing	6755
new	6623
cybersecurity	5443
infosec	4372
like	4042
security	3886
ransomware	3581
news	3566
get	3024

not finding any emerging terms and therefore performing the process becomes fruitless. While if we use a larger time period such as a month there is a higher likelihood of finding various emerging topics which may at that point have existed for some time. Therefore we found that using a period of one week provided the best result in our case. Using the TF and TFIDF processes produced two lists of relevant terms which contained within them terms relating to threats that were not currently a part of our existing filter. Two of these terms were Locky and Uiwix. Locky refers to a ransomware strain that uses email attachments for distribution while Uiwix is also a type of ransomware that uses a specific vulnerability as a worming method.

The discovery of these terms using this method shows that it can be used as a method to discover emerging and important threat names from our dataset. By using this technique on weekly blocks of data we were able to more easily track new and existing threats. A new prominent threat will appear in the lists for the first time with a high rank. That is to say, it is not as likely that a new threat will make its way up the list over multiple weeks. Since new threats are not previously public knowledge when they do emerge they will have a large amount of popularity at first meaning it will rank highly. In subsequent weeks it will often fall slowly through the rankings as posts and general discussion diminishes.

This gives us a process that we can consider when detecting new and emerging threats in our system. For our system, we use a semi-automated method that takes the TF and TFIDF based ranking on weekly subsets of our data and removes stop words and unrelated terms from the list. We also remove terms that currently exist in the word filters that are used. Since these terms already exist in our filters there is no need for them to be considered. We can then merge the two lists generated based on TF and TFIDF and highlight any previously unseen terms in the top 10. It is likely that if there is a new attack that was found or named during this time period that it would be contained in this list. This method can be used to identify the most likely terms which can be considered of interest based on their connection with the other terms already in the keyword filter.

This method is semi-automatic because the steps up until this point can be done automatically without human intervention. The final step requires a manual analysis to integrate the new terms found into the filters. This is still the case because of the nature of new threat names often being words that do not previously exist and being very creative. It is difficult to distinguish which terms are actually important and which are not. If we were to automatically incorporate the top 5 terms which are outputted using this method we may incorporate non-relevant terms and this can have a negative effect on our outputted data. Because of this, we have the final step of this process to be conducted manually to make sure it is integrating relevant terms.

3.2.3 Processing & Sentiment Classification

To implement sentiment classification we train and test various models such as logistic regression, naive Bayes, support vector machine, and decision tree. We train these models on various sentiment-based datasets to find which one performs the best in our domain. Domain-specific lexicons can give an improvement on sentiment classification in domains like cybersecurity [127]. Because of this, we test the results on classification for using a general Twitter based dataset [43] against our own cybersecurity tweet dataset. Our dataset consists of tweets mined using the previously outlined method and polarity annotated with the Python library Textblob [69]. This dataset is separate to the dataset which is used for testing the functions of the system to avoid any classification bias.

In Table 3.2 we show the classification results of training multiple machine learning models trained on a general tweet dataset to see which has the best classification score. We use bag of n-grams as the features to train our models. Bag of n-grams is a technique where we track the number of times an n-gram is used within a document. An n-gram is a collection of terms of length n such as a unigram for a single term, bi-gram for two terms, and tri-gram for three terms. Through our initial tests, the logistic Regression model provided the best results of 0.80 compared to the other models used. With further tuning, we found that using an n-gram range of 1,2 (only using unigram and bigrams) and also using our cybersecurity-specific dataset we were able to achieve an improved f1-Score of 0.88. This is shown in Table 3.3.

Table 3.2

Finally, we looked at the effect that the size of the dataset has on classification scores. In Table 3.4 we can clearly see that there is a direct correlation between the size of the dataset and the classification score achieved. As the size of the dataset set is increased there is an improvement in classification scores for our use case. This is to say if we could grow our dataset past the size which we currently have we would able to achieve improved

Table 3.2: This table shows a score comparison for various classification models trained and tested on a general tweet dataset.

Classifier	Precision	Recall	F1 Score
Logistic Regression	0.80	0.80	0.80
Linear SVM	0.79	0.79	0.79
Multinomial NB	0.77	0.77	0.77
Bernoulli NB	0.78	0.78	0.78
SGD Classifier	0.78	0.78	0.78
Decision Tree	0.72	0.72	0.72

Table 3.3: This table compares the value of using both in and out of domain datasets and the affect this has on classification score for a sentiment classification model. We found that an in domain dataset has an improvement on classification score in the case of a sentiment classification model.

Dataset	Precision	Recall	F1-Score
Created Dataset	0.85	0.84	0.84
General Dataset	0.82	0.82	0.82
Half Created Half General Dataset	0.83	0.83	0.83

Table 3.4: This table evaluates the affect that dataset size has on our sentiment classification model. Similarly to other machine learning models an increase in training data is correlated with an increased classification score

	Precision	Recall	F1-Score
79,000	0.79	0.79	0.79
200,000	0.81	0.81	0.81
400,000	0.81	0.81	0.81
800,00	0.82	0.82	0.82
1,200,000	0.82	0.82	0.82

results.

3.2.4 Visualization

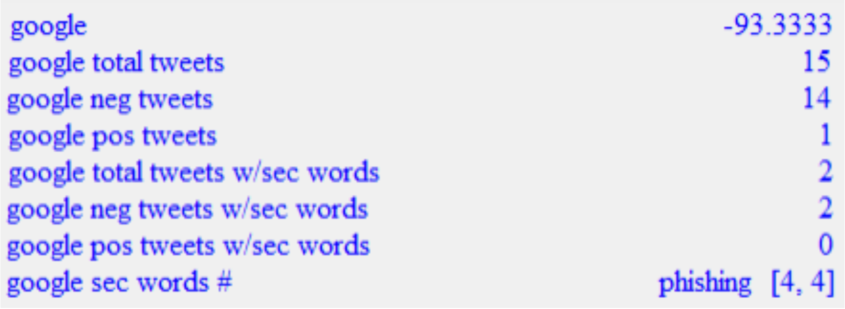
Figure 3.3 shows the main interface of our system which aggregates the general sentiment for all data which is obtained through our filtering method. In this case, it is separated based on company names within each data point to show the real-time sentiment for those companies in relation to cybersecurity data.

This method can be useful for managed service providers who handle the security of multiple organizations. It can also be useful for an organization that owns multiple brands and has various subsidiaries. In this way, they can track all of their valuable assets and judge public perception and sentiment across the whole organization.

Using this interface if there is a particular company that stands out or requires further investigation it can be selected to show greater detail about the data which represents it. In Figure 3.5 we show an example of the detailed window which can be viewed for a specific company. This shows the various number of positive and negative tweets including how many of those contain a particular attack term. As we have outlined previously, having a large amount of negative sentiment towards a company can be an indicator of a higher cyber risk level. By also looking at the particular attack words which are contained in the negative data we get a more pinpointed view of the threat that may occur. As we can see in Figure 3.5 we have a high amount of negative data some of which contains the term phishing. This information can be used by a security analyst to send an update to employees to be on guard for phishing emails or to proactively update email policies or filters for a period of time.

Use Case

Here we show a real-world use case that depicts the usefulness of our system to identify cyber threats using our framework. For this use case, we filter data based on tweets



google	-93.3333
google total tweets	15
google neg tweets	14
google pos tweets	1
google total tweets w/sec words	2
google neg tweets w/sec words	2
google pos tweets w/sec words	0
google sec words #	phishing [4, 4]

Figure 3.5: This figure shows the tweet data detail window which shows statistics about tweets captured in our system

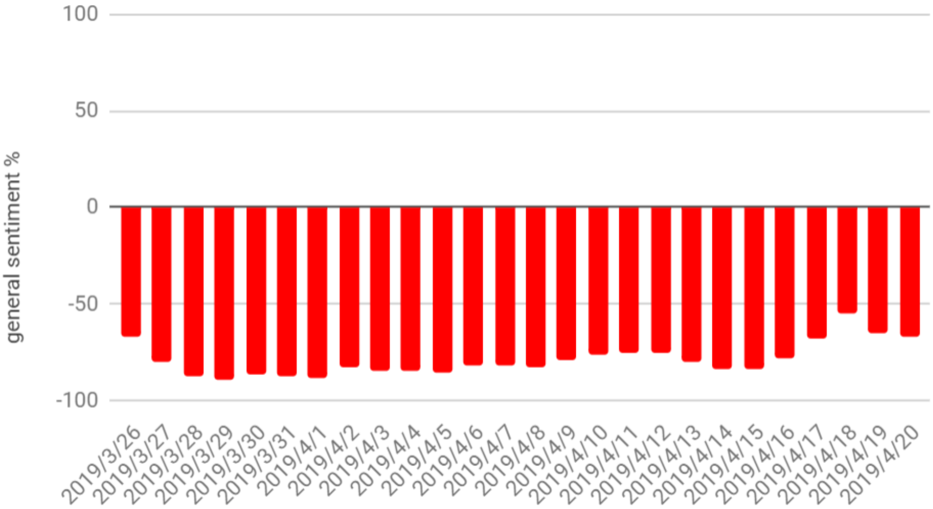


Figure 3.6: This figure shows the general sentiment values by day over the testing period for the company Google

that contain the term Google in them to reflect attacks which concern Google in some way. Because of the size and technological footprint which Google has stemming from its large amount of infrastructure and services it has a wide attack surface that attackers can search and exploit.

Figure 3.6 shows the general sentiment measurements outputted from our system in reference to Google between the dates of 2019/3/26 and 2019/4/20. Negative sentiment was recorded throughout the testing period with a low negative sentiment recorded of

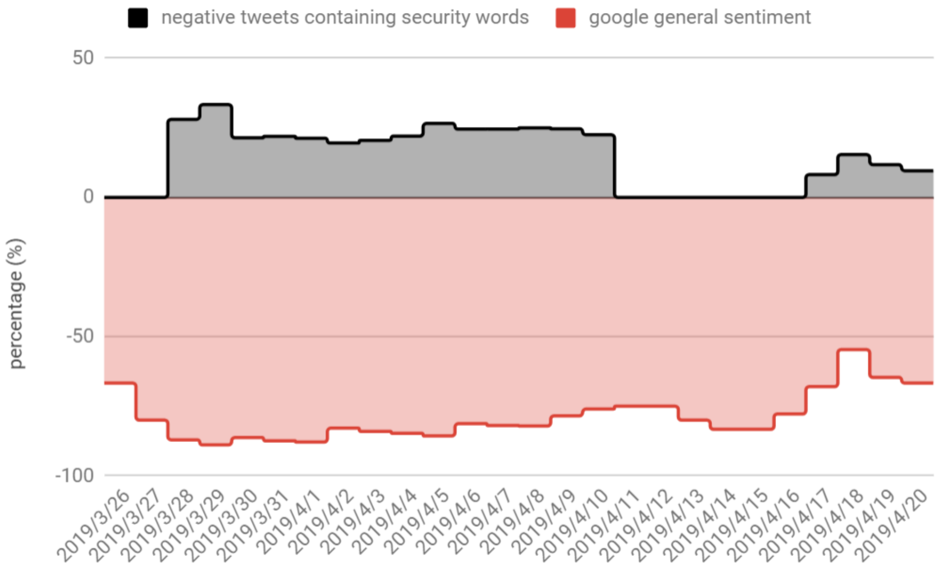


Figure 3.7: This figure shows the general Sentiment(red) vs percentage of negative tweets of total tweets(black)

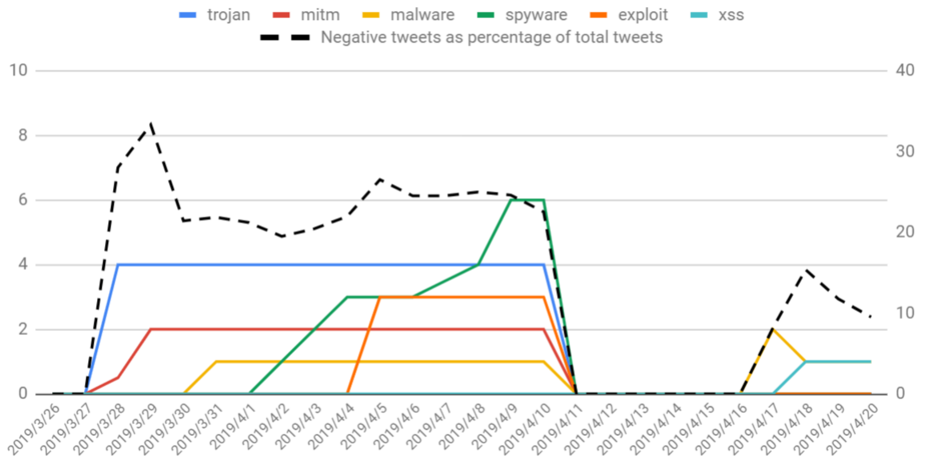
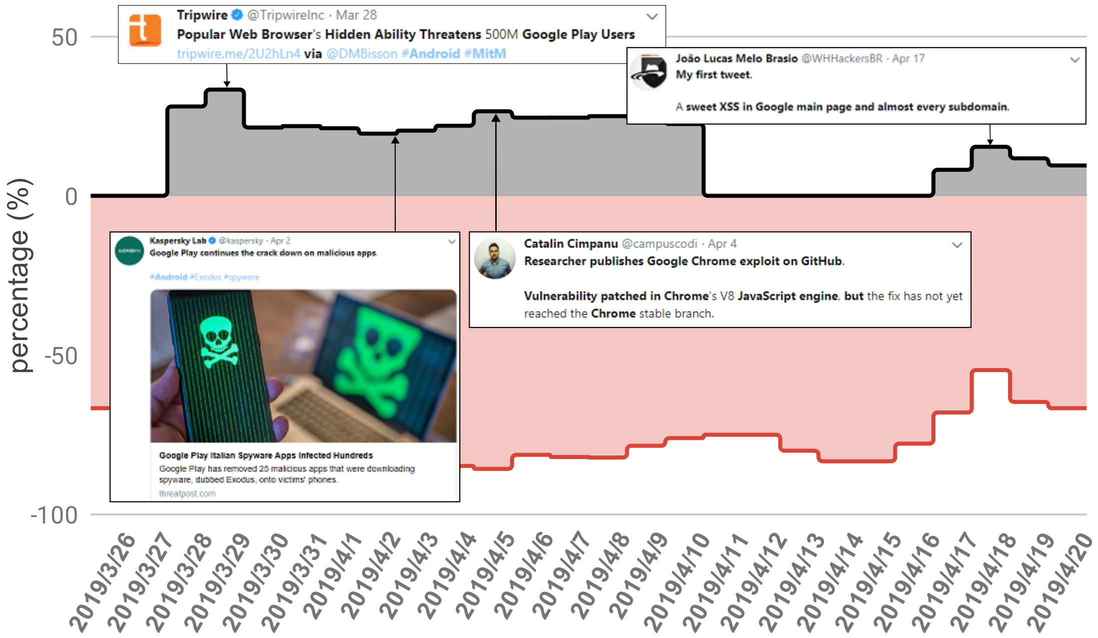


Figure 3.8: This figure measures the occurrences of specific security words in negative tweets, and how these contribute to the percentage of total negative tweets

-88% and a high negative sentiment of -55%. From this figure, we can see an increase in negative sentiment over consecutive days ending in a peak. This accumulation phase can be used as an indicator of a higher cyber risk level. In [41] they explain how analysts "would typically investigate spikes to determine what caused the feature", this matches the output of our data which shows the importance of these spikes and how they can be utilized by an analyst. Analysts can gain further insight into these spikes by looking at the details and especially by viewing how many negative tweets contain attack terms.



Tripwire @TripwireInc · Mar 28
Popular Web Browser's Hidden Ability Threatens 500M Google Play Users
tripwire.me/2U2hLn4 via @DMBisson #Android #MitM

João Lucas Melo Brasio @WHHackersBR · Apr 17
My first tweet.

A sweet XSS in Google main page and almost every subdomain.

Kaspersky Lab @kaspersky · Apr 2
Google Play continues the crack down on malicious apps.

#Android #Exodus #spyware

Google Play Italian Spyware Apps Infected Hundreds
Google Play has removed 25 malicious apps that were downloading spyware, dubbed Exodus, onto victims' phones.
threatpost.com

Catalin Cimpanu @campuscodi · Apr 4
Researcher publishes Google Chrome exploit on GitHub.

Vulnerability patched in Chrome's V8 JavaScript engine, but the fix has not yet reached the Chrome stable branch.

Figure 3.9: This figure shows examples of specific tweets which can be found using the peaks of our negative tweet indicator to gain insight into specific attacks

Figure 3.7 shows the ratio of total data which is taken up by negative sentiment posts. In a similar way when we have days where a higher percentage of the total data is being taken up by negative sentiment posts we can use this as an indicator.

We have mentioned the importance of gaining more detailed information about a specific event to better understand its cause. In our case, this is done by viewing the occurrence of attack terms within negative sentiment tweets in our data. In Figure 3.8 we can clearly see the peaks which are created by the increase in specific attack terms and how they affect the overall negative sentiment.

In the case of the first peak on 2019/3/29, it arises with the occurrence of the word trojan and then further jumps with MITM (Man in the Middle). This was in relation to a suspected malicious browser application that was hosted on the Google Play Store [1]. This browser had included a functionality that allowed it to download potentially unwanted libraries onto the user 's device [22]. As discussion increased in regards to this threat there was a change in the words being used in relation to the attack. The introduction of the reference to the same threat as a MITM attack shows how the discussion for threats can be pushed by re-categorization. By discussing the same threat as a new type of attack it can push awareness of that threat. This can be because of new details or functions found by researchers or a miss-categorization by new media outlets.

For the second peak on 2019/4/5, we can see that the rise correlates with the word spyware and then is exacerbated by the rise of the word exploit. This spike occurred in conjunction with "Exodus" spyware which was also hosted on the Google Play platform. Exodus was spyware that imitated a legitimate application to steal sensitive information from its users on the Android platform [120]. This spyware was principally found in Italy and therefore posed a greater threat to Italian users. With this information, security analysts for italian companies or security-conscious italian users could better protect themselves by being aware of the existence of this spyware and not downloading it [115].

Finally, the last peak is created by the occurrence of an XSS attack. In this case, it came from an individual tweet from a security researcher who discovered an XSS vulnerability on the Google website. This particular example shows the occurrence of a security vulnerability that is present in a large company that did not gain widespread media attention and therefore would have been very difficult to identify manually. Because of the number of vulnerabilities that exist many are not widely reported about and therefore can be missed by people who learn about vulnerabilities in this way. Using our system we can use this indicator to identify both threats which are widely discussed and those from individual researchers that do not meet mainstream popularity.

This shows some of the benefits that can be gained using our system. It can allow an analyst to be much more effective by being able to effectively process large amounts of

[1] https://play.google.com/

data which otherwise would not be possible. This allows them to become more aware of specific threats and vulnerabilities like the examples shown previously. When we consider the traditional manual methods which were commonly used by analysts this method is able to achieve a substantial improvement in comparison. This manual technique is well described by a line in [41], "From these reports, he identifies a list of approximately 50 threats that he needs to examine that day. He prioritizes this list to determine the top 10 that must be addressed. Of these, he is usually able to tackle the top 3-5 during the day."

This type of analysis may have held benefits prior to the explosion of social media and data-driven services, but considering the amount of data available today, our method is an appropriate solution. Considering the amount of time needed to analyze and triage a large number of articles, our system can be leveraged to do this task in a fraction of the time required. This is especially important within the cybersecurity field where time can be crucial in stopping an attack. To show a comparison between previous methods and what can be achieved from our system we look at the number of Google searches performed for the threats we have been able to identify through our system. Considering the wide use of the Google search engine we can correlate the number of searches for these threats, with users who found out about these attacks by searching to find further details. By comparing this with our results we can understand the difference between using a manual method of discovery and using our system.

In Figure 3.11 we show a comparison of Google searches performed during the same period as the attacks highlighted by our system. These Google searches contain the specific words used in those attacks to represent how popular these searches were. When looking at the first attack we outlined in "UC Browser" our system was able to detect it three days before the general awareness of this threat became widespread and also one day before the associated vulnerability was published in the NIST vulnerability database (CVE-2019-10251). Similarly in the case of Exodus malware our system was able to detect posts of this threat one day before it was widely discussed through the use of our negative sentiment measurement which peaked during this period. In the case of our third example of an individual researchers tweet, this was not discussed widely showing a crucial point in that our system can assist in the discovery of threats and vulnerabilities which are not widely discussed and therefore harder to become aware of.

To further demonstrate the benefits of our system in comparison to a manual method, we found that when looking at popular cybersecurity websites, our system was able to detect posts related to the featured threats before the website posted an article about the same threat 87.3% of the time. In this way, we can ensure that using our method users can become aware of threats and vulnerabilities and gain an overall better view of the threat landscape faster and clearer than manual methods. This allows users to focus on the priority task of implementing patches and fixes for these threats in a timely manner before they can be negatively affected by them.

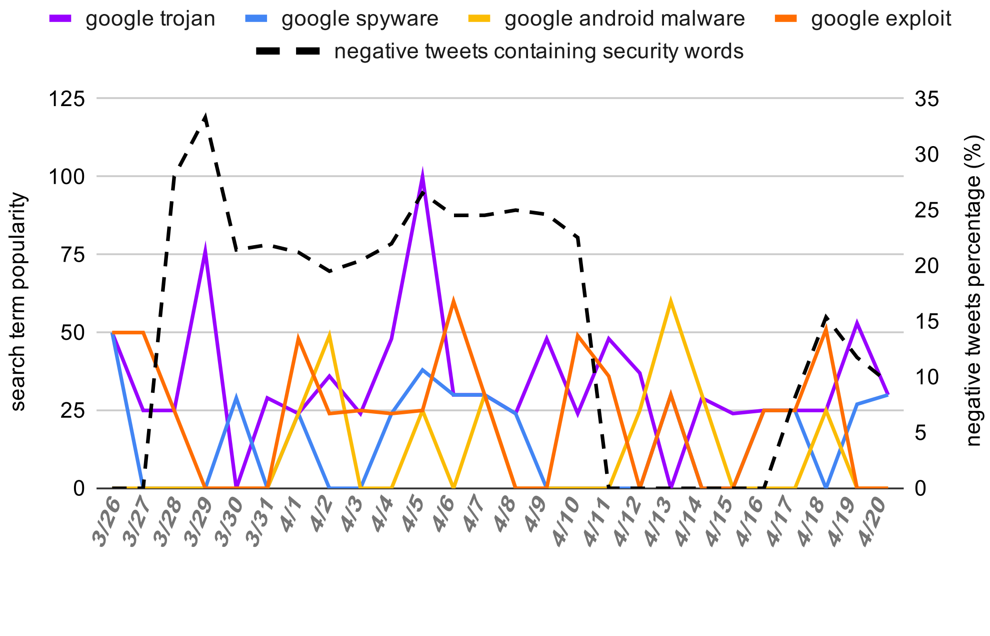


Figure 3.10: This figure shows a comparison of our indicator take from negative sentiment tweets, compared against Google search popularity for the same terms which we tracked

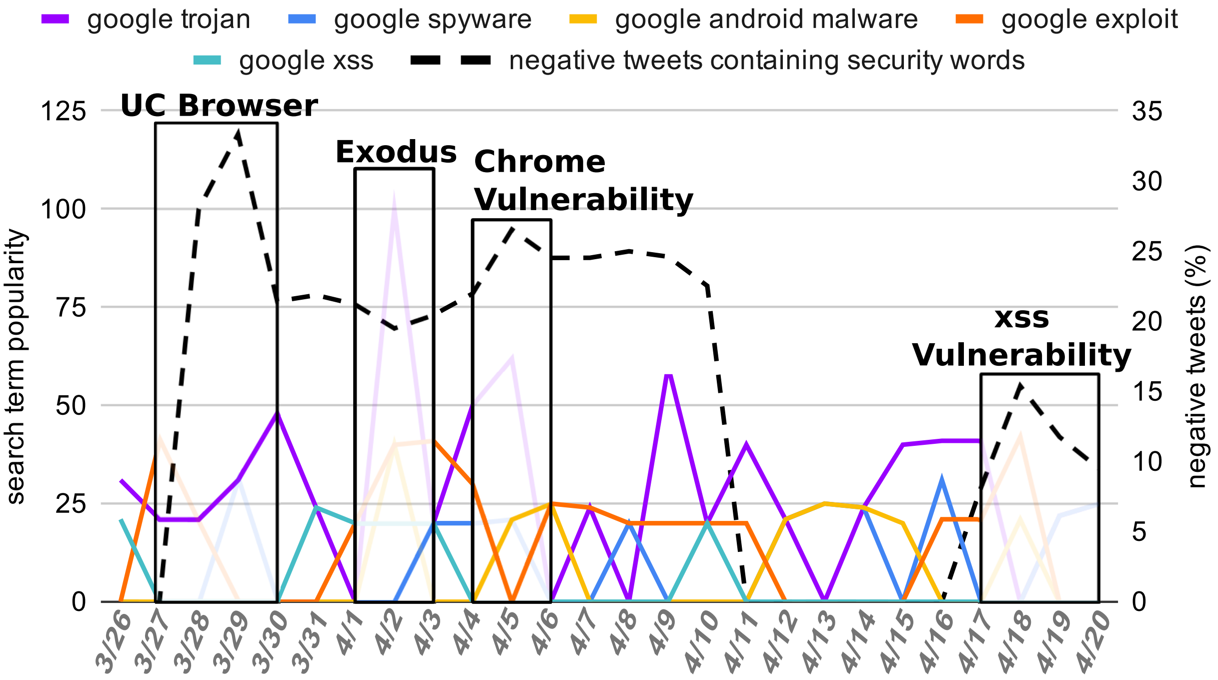


Figure 3.11: This figure shows our negative tweets measurement method has greater sensitivity to attacks and is able to detect threats quickly. This can be used as a preliminary indicator of a new or emerging attack

Chapter 4

Enhancing Cyber Threat Intelligence Data Quality

Previously we presented our framework for a CTI system using open-source data through Twitter posts and uses data mining techniques to extract useful information and insights into the data. In the process of developing, implementing, and testing this framework we found key components that are crucial to the effective running of this system.

From these insights, we found that the most critical component of this type of system is the data itself. The data acquisition phase when data is being filtered and a decision is being made as to which data points to integrate into the data stream is crucial to the reliable output of a CTI system. This is represented by the common computer science concept of ”garbage in, garbage out”. This phrase refers to the concept that flawed, erroneous or generally bad input data can be expected to produce unusable and unreliable output. Therefore this component of a CTI system needs to be considered carefully to establish its accuracy and the reliability of the whole system.

For this reason, we present a method that concentrates on filtering and extracting relevant and useful data from a general data stream of multiple open cybersecurity data sources. This is done with the goal of enhancing the quality of data inputs that go into a CTI system by filtering out non-relevant and non-significant data.

4.1 Method

4.1.1 Dataset

Table 4.1: This table shows an overview of the datasets used in this portion of the study showing the size of each dataset (Total-Entries), the size of the training and testing sets for each dataset (Training-Set, Testing-Set), the data makeup of the training and testing sets showing how much security-related (positive) and Non-security related (negative) data is present, and the average length of a data point in that dataset (Avg Length).

Dataset	Training-Set	Testing-Set	Total-Entries	Training-Set Class Distribution	Testing-Set Class Distribution	Avg Length
Full (F1)	1,016,346	179,930	1,196,276	50%Security 50%Non-Security	50%Security 50%Non-Security	607
Twitter (T1)	139,841	24,674	164,515	53%Security 47%Non-Security	53%Security 47%Non-Security	90
Reddit (R1)	9709	1713	11,422	50%Security 50%Non-Security	50%Security 50%Non-Security	660
SE (S1)	858,801	151,549	1,010,350	50%Security 50%Non-Security	50%Security 50%Non-Security	935
Articles (A1)	7996	1995	9991	50%Security 50%Non-Security	50%Security 50%Non-Security	3807
Short (S2)	313,839	34,582	348,421	63%Security 37%Non-Security	41%Security 59%Non-Security	86
Medium (M1)	611,892	117,584	729,476	45%Security 55%Non-Security	59%Security 41%Non-Security	3070
Long (L1)	90,612	27,761	118,373	41%Security 59%Non-Security	65%Security 35%Non-Security	513
Annotated (A)	na	500	500	50%Security 50%Non-Security	50%Security 50%Non-Security	1125

In Section 3 we presented the framework for our CTI system, during the implementation and testing of this system we used Twitter data due to its widespread use throughout the cybersecurity community and its function of acting as a news aggregator by hosting information from other platforms. As has been mentioned, the ideal scenario for a CTI system is to be able to retrieve a wide scope of data from various different platforms to ensure the greatest likelihood to retrieve relevant and useful information. Because of this, to test our MLKF+C method we have used data from multiple platforms that contain differing intrinsic features.

Posts can differ greatly between platforms, while a Tweet is short, unstructured, and noisy, an article can be long, structured, and organized. Because of this, we separate our data into specific feature-based datasets based on platform and length. The inherent features which are present in various platforms can have a large impact on the techniques we use and the effect each technique has on a data point. By performing these tests on individual subsets of our data we can track the particular differences between them and judge how to proceed. This has various benefits when it comes to the practical application of our methods in a CTI system. By knowing which type of classifier is best under certain feature sets we can optimize our models for specific datasets. This allows us to target our ML models to optimize their results based on the target use case such as greater classification.

In our results, we show how all tested models score for each subset of the data to outline the benefits and drawbacks of our method and other similar methods which allows us to get a realistic view of what can be gained from this system. In Table 4.1 we can see the various statistics for our datasets showing the sizes used for the training and testing sets as well as the various class distributions based on positive and negative data.

4.1.2 MLKF algorithm

To achieve this we implement two principal methods. Firstly we develop a custom filtering algorithm called the multi-layer keyword filtering which expands upon keyword filtering. Keyword filtering is a common technique used in areas such as search engines, parental control systems, and other filtering methods. Even Though keyword filtering is a basic technique it can be applied across any domain quickly and effectively. With the increased need for training and testing datasets for use in ML experiments, many researchers have turned to use this technique to automatically create their datasets. This is done due to a lack of gold standard datasets in cybertext analytics forcing researchers to create their own datasets. Keyword filtering is a natural choice in these situations due to its ease of implementation and relatively good results.

At the same time keyword filtering has issues with false-positives. It is not able to disambiguate between terms and therefore is not able to know the context in which a term is used, because of this it can incorporate non-relevant data into a dataset or in our case into a CTI system. We expand upon keyword filtering by creating an algorithm that adds context detection to it through an associated words list. Our method takes keyword filtering and adds two more layers to it. These layers further filter the data depending on the context of the term to the related domain. The first layer consists of a double meaning word filter (DMWF) which lists all the terms in the main filter which can be used in a different context or have a use outside of the cybersecurity domain. In this way, we can identify posts containing these terms for further inspection.

Once we have identified posts that contain a term from the DMWF we go through a process to find out whether the post should be discarded due to not being relevant or if it should be stored. Firstly, we look at the number of terms from our DMWF which are found within the post. Although the terms from this list are ambiguous individually when used in conjunction with other similar terms the likelihood of the post being relevant rises greatly. Therefore if more than one term from the DMWF list is found in the post it is considered relevant and stored. If there is only one term from the DMWF found in the post we go on to further process it. In this case, we go on to use the term in question as the seed term for a word2vec generated word list. The associated words list (AWL) is a list of the top terms which are related to the seed term in a cybersecurity context. This list includes the most common terms which are found with our seed term when it is used in this context. By scanning the post to look for terms contained in the AWL we can make a decision about the relevance of the post and either drop it or store it. In this way, we are able to add a level of context to keyword filtering to reduce the number of false-positives and obtain higher quality data. This method has various parameters and specific techniques that affect its usefulness and its inner workings. In the next section, we go into detail about the various aspects which affect this algorithm and how to get the best results from it.

Algorithm 2 Pseudocode for multi-layer keyword filtering algorithm for classifying cybersecurity data.

```
procedure CLASSIFYING TEXT DATA AS CYBERSECURITY RELATED OR NOT
    Main_Filter ← Complete keyword filter containing all terms
    Double_Meaning_Word_Filter (DMWF) ←
    Keyword filter containing terms that have two meanings
    Associated_Words_List (AWL) ←
    Dynamically generated list of similar terms based on input term

    Main:
    if If post does not contain any term from Main_Filter then
        Discard
    else
        if post contains a term in Double_Meaning_Filter then
            if greater than one match in post and Double_Meaning_Filter then
                Store
            else
                generate Associated_Words_List using word2vec model
                remove original Keyword from the post
                if post contains Keyword from Associated_Words_List then
                    Store
                else
                    Discard
        else
            Store
```

4.1.3 MLKF+Custering algorithm

The first step in our goal was to ensure we are retrieving a stream of relevant data to establish the reliability of our system. To further enhance the quality and output of a CTI system we have to not only consider the relevance of the data, but we must also consider the significance of the data points to make sure that they are significant to our goal. Post significance can be considered based on how much contribution, insight, or context a particular post is able to give for a particular attack or threat. Posts that can give some type of insight into a threat would obviously be considered more significant than posts that do not provide any benefit in this realm. Moreover, posts that do not provide any insight or context into threats can negatively affect the system and should be removed. Posts that would be considered less significant are device and product marketing and advertising posts, service promotion, and general posts about security conferences, exams, and certification. In these cases processing these types of posts (especially in a large number) can affect the way the overall data is viewed from a CTI system and cause unreliable outcomes to be considered.

Therefore to further improve our algorithm we have added a clustering component to it called multi-layer keyword filtering + clustering (MLKF+C). This clustering component uses a clustering algorithm on positively classified data points from our MLKF algorithm and groups them to find the underlying topic of the post. By clustering these posts based on the topic we can identify which groups contain a larger amount of non-significant posts such as advertising and marketing data. In this way, we can gain an insight into the distribution of significant data at a particular time, and we can decide whether to remove certain clusters to increase the significance and usability of our overall dataset. Similarly, there are various parameters and measures which we must take into account when judging the efficacy of a clustering method. Unlike our previous method, we do not have an annotated dataset to evaluate the topics of our clustering algorithm. We therefore look at various measurements that judge the effectiveness of clustering based on cluster size and distance. We also manually evaluate various clustering algorithms to determine the effectiveness with which they can identify non-significant groups within our data.

By employing these two methods we are able to improve upon base keyword filtering while still retaining its applicability and flexibility. This adaptation in creating the MLKF+C method has presented a way to enhance the quality of a data stream by filtering out non-relevant and non-significant data and condensing the data down to an accurate representation of the attacks and threats which we want to gain more insight and context on.

4.2 Results

During the development and implementation of the MLKF+C algorithm various testing and investigatory work had to be conducted to gain the proper insights into how the algorithm handles and under what conditions it works best. To gain this insight we performed examinations on each facet of our approach and were able to gain detailed insights into the various areas of its applicability. In this section, we go into detail on each tested component and how its tuning and configuration affect the greater outcome.

4.2.1 Data Stemming and Lemmatization

We talked about the uses of stemming and lemmatization to reduce terms down to their common base form. This can have various effects in our use case which must be considered. The main reason to use stemming or lemmatization is to reduce our vocabulary size and get some benefit in classification score. In our algorithm, we are developing a method that can better identify non-relevant cybersecurity text posts. By implementing either stemming or lemmatization we are effectively increasing the number of ambiguous terms in our data by reducing terms down to common shorter versions. This may not be beneficial to methods that have trouble with false-positives and do not implement any extraneous method to deal with it but in our case, because we implement a context disambiguation

Table 4.2: This table shows the classification scores achieved for our method when using root form generation such as stemming, lemmatization, and leaving terms in their original form.

	F1-Score			**Vocab Size**		
Dataset	**lemmatization**	**none**	**stemming**	**lemmatization**	**none**	**stemming**
F1	0.7769	0.7585	0.923	613,264	625,542	560,823
T1	0.2773	0.9972	0.9993	55,269	59,222	48,039
R1	0.8937	0.8327	0.9245	38,253	42,268	29,909
S1	0.8499	0.8476	0.9252	555,488	565,216	509,906
A1	0.8801	0.9198	0.9731	66,489	73,238	52,179
S2	0.2951	0.9649	0.9969	79,340	84,508	67,668
M1	0.8294	0.8018	0.8898	285,628	294,459	251,758
L1	0.8982	0.8998	0.9251	423,369	433,236	389,508

layer increasing the number of ambiguous terms is not an issue but preferred.

In Table 4.2 we can see the results from using stemming, lemmatization, and the original token. We show the results in the form of f1-Score from dataset classification and the final vocabulary size of the dataset once processing was conducted with the particular technique. From these results, we can see that stemming provides the best results in all cases. By using the stemming technique we are able to achieve a smaller vocabulary size which raises the number of ambiguous terms in our dataset and optimizes data pre-processing for our use case. This smaller vocabulary size has a correlation with the f1-Score as we can see since the stemming method results have the smallest vocabulary sizes in all cases and highest f1-Scores. While the original tokens without any processing all have the largest vocabulary sizes and lowest f1-Scores. The addition of a processing step can add extra processing time especially in the case of lemmatization which uses a dictionary to lemmatize terms. By using stemming instead of lemmatization this extra processing time is reduced and is outweighed by the improvement in classification score.

4.2.2 MLKF Parameters and Configuration

The MLKF algorithm we have developed has various intrinsic and acquired components that can be adjusted to change the internal functions of the algorithm. Within our algorithm, we use the word2vec model to generate an AWL. The terms which are generated by this list are based on the training data which is used to train the model. To ensure the model can effectively represent the similarity of words from its training set, two principal hyperparameters can be changed to control the number of training iterations that the model goes through and the size of the vectors used. In our testing we look at the effect of different epoch and vector size values on our AWL generation and how these settings affect the classification score.

Furthermore, the length of the AWL is a parameter that has been gained from the development of this algorithm. The length of the AWL can affect classification by controlling the margin of error when classifying posts. The length of the AWL list effectively allows us to have a dial to regulate the strictness or looseness of false-positive acceptance in our algorithm. By having a longer AWL we will effectively accept more posts as relevant which contain a greater range of terms and by having a shorter AWL our algorithm will become more strict in its classification. We look at these two configurable options and how various values affected our algorithm's classification score as a whole and under specific features.

4.2.3 Word2vec Epochs and Vector Size

Here we look at the number of training epochs used to train the model and the length of the vectors produced in each embedding with the aim of finding the optimal settings for our use case. We tested epoch ranges and vector sizes from 10 to 100 and show the best results in Table 4.3.

Looking at the f1-Score results we can see that when we see shorter documents, either in terms of length or size, higher epochs are needed to extract more context. This can be

Table 4.3: This table shows the vector size, training epochs, and associated words list length which produced the best classification scores for each dataset on both f1-Score and precision metrics.

	Vector Size		**Epochs**		**AWL Length**	
Dataset	**F1-Score**	**Precision**	**F1-Score**	**Precision**	**F1-Score**	**Precision**
F1	50	10	50	50	2500	0
T1	10	10	100	50	2500	0
R1	10	50	50	50	2500	0
S1	50	10	50	50	2500	0
A1	10	10	50	50	2500	0
S2	50	10	100	50	2500	0
M1	50	10	100	50	2500	0
L1	50	10	50	50	2500	0

seen in the T1 and S2 datasets which utilize 100 training epochs. Smaller datasets benefit from a smaller vector size as we can see in the T1 dataset vector size of 10. Therefore, in the event of having a smaller dataset that does not contain a large enough amount or a variety of data the training epochs can be increased and we can use a smaller vector size to make our embeddings more representative. Although the increase in training iterations can be used to extract more context this only works up until a point, after this point we begin to see diminishing returns. This is represented in our method with the T1 dataset where precision falls as we increase vector size past 10, and epochs past 100 meaning our model has begun overfitting to our data.

In documents that have a larger vocabulary size, we can see that 50 training epochs produce the best results, half the amount used on shorter datasets. This is accompanied by vector sizes of either 10 or 50. In the case of datasets that have a longer length such as R1 and A1 both had a vector size of 10. Due to their long length, these texts were able to be better represented in smaller vectors. In the cases of M1, S1, and L1, they had a longer average length but did not have the same effect. This is most likely due to platform-specific factors such as code and markup language contained within the data point. This shows a clear distinction between the long text which contains programming code and other unused text and structured articles which are used in their entirety.

In cases where the unused text was present the average length of the documents in these datasets was high before being processed, but after processing the length of the document lowered affecting the results. For this reason, aggregated datasets containing code such as L1, S1, and F1 use training epochs of 50 as well as a vector size of 50. Although the precision metric is not being used as a measurement of effectiveness for our system it still is able to give us insights into how our algorithm functions. Looking at the precision score we see that all cases other than R1 perform best when using a vector size of 10 and 50 training epochs. This is because the precision metric maximizes the number of true-positives by not attempting to process certain data-points which will increase the false-positive score. By ensuring the highest amount of true-positives, more data will be correct but the false-negative count will increase.

The ideal situation is to have a large amount of training data available to train our model but in practice, this is not always feasible. For this reason, it is important to find optimal settings in various types of situations where there is enough data, and when there is a smaller amount. The results we have achieved here are specific to our domain and our data. There is no standard length or optimal setting across all domains our tests are able to show specific and accurate insights into our data for our use case.

4.2.4 AWL Length

Figure 4.1 shows the effect of our AWL generated by the word2vec model on the classification scores in relation to the length of the generated list. Firstly, we can see that the f1-score metric shows an improvement while the precision score shows a mirrored decline in classification score. As previously mentioned this is due to the precision metric max-

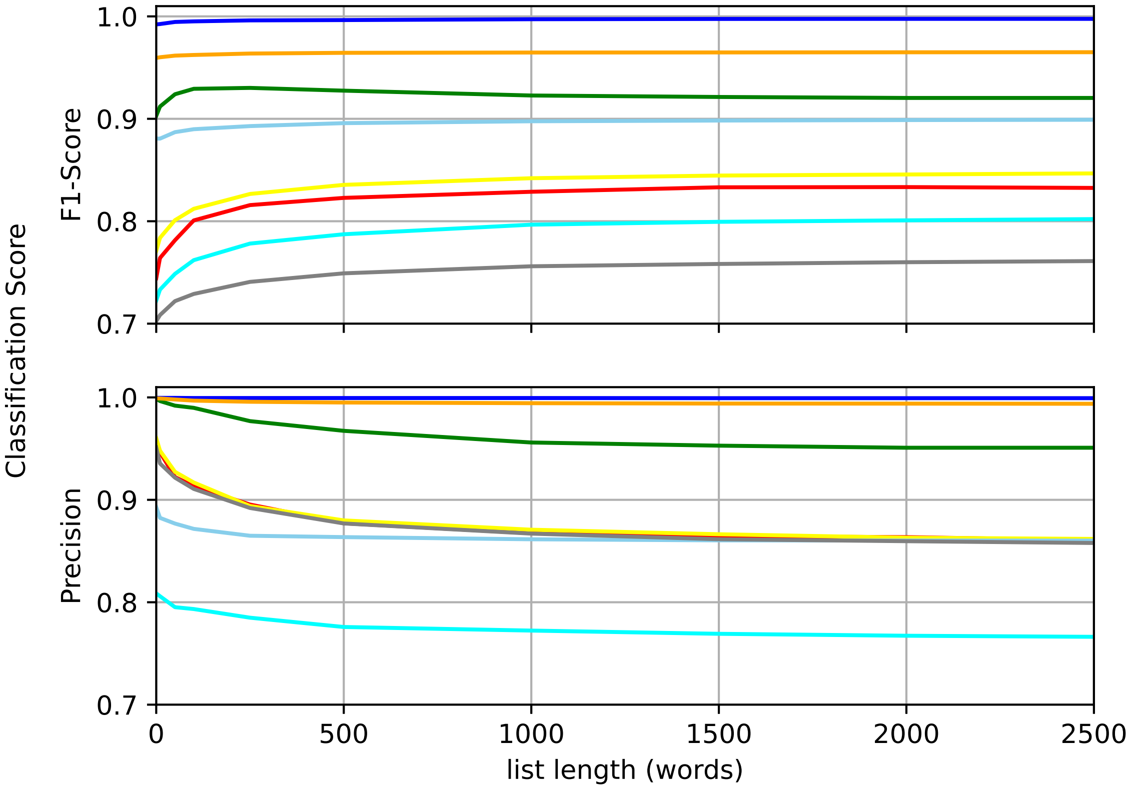


Figure 4.1: This figure shows the affect that the length of our associated words filter generated by our word2vec model has on classification for each dataset.

imizing true-positives. The precision metric performs well when it does not process any ambiguous terms and therefore does best when the associated words list is at 0. By not processing ambiguous terms it does not risk introducing false-positives and maximizes its precision score. This is not beneficial for a CTI system in a practical sense since a large number of terms in the cybersecurity domain are ambiguous and leaving out this data dramatically reduces scope in a CTI system. For this reason, we analyze the f1-score results which give a more balanced representation of classification since it also takes recall into account.

In Figure 4.1 our f1-scores are grouped by dataset size. The S1 and T1 datasets for

Table 4.4: This table shows the change in classification score for various list lengths, in most cases the greatest increase in classification score is achieved in the 0 to 50 and 0 to 250 categories.

Datasets	**0 - 50**	**50 - 250**	**250 - 1000**	**1000 - 2500**	**0 - 2500**
F1	1.9564	1.8802	1.5220	0.5071	5.8659
T1	0.2377	0.1528	0.1142	0.0342	0.5390
R1	3.7120	3.4316	1.3006	0.3767	8.8211
S1	2.9976	2.5613	1.5370	0.4693	7.5654
A1	2.1182	0.6200	-0.742	-0.238	1.7571
S2	0.2271	0.2033	0.0980	0.0318	0.5607
M1	2.6051	2.9592	1.8458	0.5441	7.9545
L1	0.6238	0.5919	0.4707	0.1479	1.8345

short length data which have the best results are grouped, followed by the L1 and A1 long length dataset, and the medium-sized datasets S2, R1, M1, and F1. We can see that the groupings show similar classification trends where datasets in the same grouping have similar peaks, showing optimal list length. This can be seen most clearly in S1 and R1 which are from similar platforms showing related classification trends, this is also the case for the M1 medium length dataset and the F1 full dataset which follow the same trend. The F1 full dataset follows a similar trend but it is not as distinctive as the medium length datasets due to the fact that the full dataset has a greater percentage of medium length entries. We see that our AWL has the most impact on medium length datasets with a 6.4% increase for S1, 7.8% increase for R1, and 6.4% increase for M1. These gains are all achieved at the list length within 500. While the F1 dataset has a 4.6% increase within 500 terms. For the short length datasets, there is a less significant impact with a 0.3% increase in 100 words for the T1 dataset and the S2 dataset. The reason for this being the smaller document sizes are less likely to find correlating terms within our AWL.

In the case of the A1 dataset, we see a good example of a point of diminishing returns with the growth of the AWL. The A1 dataset achieves its best results at a length of 250 terms with an increase of 5.6%. After this point, we see a gradual decrease in classification score as the list length increases with a total 3% loss from its peak to the maximum length of 2500. If we contrast this with the L1 dataset, it also peaks at 250 words and increases by 1.2% at a maximum length of 2500.

The A1 dataset was negatively impacted as the length of the AWL grew with a decline in the 250 - 1000 and 1000 - 2500 categories which shows the change in classification score based on the range of terms in the AWL for the f1-score metric. Ultimately all datasets were positively affected by this method with the classification score being higher at the maximum length of 2500 than it was at 0. The most beneficial impact from our AWL came within the first 500 terms with medium-sized datasets benefiting the most.

Medium-sized datasets showed the greatest increase since both related and unrelated documents in these datasets are similar compared to other platforms. These datasets are comprised of Reddit and Stack Exchange posts which are often technical in nature and therefore have a larger quantity of ambiguous terms to process. This shows that our method can help in the identification of documents which are in tangential domains but not in cybersecurity specifically. This technique had less of an effect on shorter length datasets, with datasets being smaller in size and vocabulary there is less likelihood of a term being matched in the filter and thus reduces its effect. Also in contrast to the medium-length datasets, Twitter has a wider scope of conversations with less ambiguous documents which can benefit from this step in our method. Even though this is the case this type of method is intrinsically suited to short length data and because fo this achieves its best results for those datasets.

The A1 dataset shows a peak in classification score followed by a gradual decrease. This is due to the long length of articles, by increasing the length of our filter there will be more matches, which increases the likelihood of false-positives as is shown in Table 4.4.

Even though this is the case there is still a benefit to classification by having a shorter length list for this dataset which achieves a 2.1% increase. There is not a similar trend in the L1 dataset because it contains a large number of entries from R1 and S1. As we have shown these platforms benefit more from this type of filter and thus results for the L1 dataset plateau, if this dataset contained more entries towards either platform it would be expected to exhibit the same effects that we can see in their respective groupings.

4.2.5 MLKF Classification

Based on the previous tests done we were able to find the optimal combination of parameters for our MLKF method in terms of classification score. By applying this configuration to each particular dataset we can find the f1-Score for each dataset and how it performs. In table 4.5 we present the classification scores for our MLKF method compared with the baseline methods of naive bayes, logistic regression, and stochastic gradient descent models. We also compare our algorithm against a state of the art model in BERT.

The MLKF algorithm performed best with short-length datasets, particularly the T1 and S2 datasets. In the case of the S2 data, it achieved the highest f1-Score with 0.996 which was 0.079 higher than the closets model. In the T1 database, an f1-Score of 0.999 was achieved which was effectively equal with the BERT model. This shows that our method performs best on short text formats which will often have smaller vocabulary sizes. The T1 dataset is made up completely of tweets while the S2 datasets has a large amount of tweet based data. While tweets are obviously short in length they can have a large vocabulary size due to the amount of misspelled words creating unique tokens. We have shown that this can lead to a worsened classification score, but by using stemming in conjunction with our AWL we are able to reduce this effect and get a high classification score.

This method performs less well on the M1, S1, and R1 datasets. These datasets are generally of medium-sized articles and posts which largely come from the Stack Exchange platform. The keyword filtering method is inherently less suited to perform well on

Table 4.5: This table shows the classification results on all of our datasets using the f1-score and precision metrics for our method, other baseline methods and BERT.

Dataset	Metric	MLKF	BERT	NB	LG	SGD
F1	Precision	0.8586930	**0.987722**	0.9291108	0.9155384	0.9304398
	F1-Score	0.9230412	**0.970297**	0.9288186	0.9284404	0.9034246
T1	Precision	0.9992395	**0.999848**	0.9510397	0.9995433	0.9998474
	F1-Score	0.9993915	**0.999393**	0.9686385	0.9969634	0.9960477
R1	Precision	0.8596273	**0.965311**	0.8344965	0.9399249	0.9522058
	F1-Score	0.9245156	**0.953337**	0.8999999	0.9070048	0.9288702
S1	Precision	0.8622018	**0.946769**	0.9131649	0.9163208	0.8815229
	F1-Score	0.9252420	**0.932649**	0.9222386	0.9127822	0.8684897
A1	Precision	0.9497326	**0.985872**	0.9191530	0.9773429	0.9867482
	F1-Score	0.9731506	**0.982403**	0.9376534	0.9639410	0.9782718
S2	Precision	**0.994649**	0.8666085	0.6674817	0.8023202	0.7309531
	F1-Score	**0.996983**	0.9173368	0.7986046	0.8854578	0.8391003
M1	Precision	0.8118452	**0.967056**	0.9436900	0.9422468	0.9423102
	F1-Score	0.8898896	**0.940931**	0.9204483	0.9195908	0.8912031
L1	Precision	0.8609181	0.9478483	0.9852198	0.9863325	**0.987827**
	F1-Score	0.9251866	0.9401587	0.9367879	**0.957801**	0.9511177
A	Precision	**0.983606**	0.9510204	0.95	0.9497907	0.9681818
	F1-Score	**0.991735**	0.9433198	0.9325153	0.9303278	0.9083155

these types of datasets based on the lexical similarity between positive and negative documents and often containing unused data. Having said this our method provides a notable improvement over basic keyword filtering as was shown in the previous section by removing false-positives. In this way, MLKF is able to achieve f1-Score which is comparable to BERT in the R1 and S1 datasets for a data type that it is not suited for.

When looking at the datasets which contain a longer average length such as L1 and A1 we can see a good example of how unstructured text can have an effect on classification score using the MLKF method. In the case of the A1 dataset which is composed of traditional news articles that are well structured and do not tend to contain spelling or grammar mistakes, we are able to see an f1-Score of 0.97. Even Though our method is not suited to longer document sizes it is able to effectively classify these types of documents due to their structured nature. In the L1 dataset, we can see an f1-Score of 0.92 which is comparatively less when looking at the results of A1. This shows the importance of the pre-processing stage on unstructured texts. By using effective pre-processing methods on unstructured documents that are able to retain the underlying relationships and meaning of the original data, we can achieve positive results with documents of all lengths as we have seen in the A1 dataset.

The BERT model achieved the highest results on multiple datasets such as the F1, T1, S1, R1, A1, and M1 datasets. In the case of the T1 dataset, our method and BERT essentially have the same classification score with it being within .0001% of the BERT model. BERT is a state-of-the-art model that benefits from using transfer learning for classification and therefore would be expected to do well in these classification tasks. This shows that transfer learning can be used successfully to classify cybersecurity based posts and articles, particularly for medium-sized texts. Even though this is the case within this research we are looking at the applicability of these classifiers within the realm of their appropriateness and applicability for a CTI system which we will look at in the next chapter. In Table 4.5 we can see that the BERT method does not do as well on the L1 dataset. As previously mentioned this is influenced by the unstructured and noisy characteristics of the documents but is also be contributed to by BERT's tokenization step. BERT must truncate document sizes to a maximum of 400 characters which caused the documents in this dataset to be cut short and hence context and valuable testing data removed. Because of this, in the case of the L1 dataset, the baseline methods achieved the best results.

These results show the effectiveness and usability of the MLKF algorithm in classifying documents from different platforms and containing various feature types. Even Though we are able to identify datasets in which it obviously contains inherent benefits and therefore produces better results in all cases except for the L1 dataset we were able to achieve an F1-Score above 0.90 with very good classification scores in short texts.

4.2.6 MLKF+C Topic Clustering

In this phase, we add a clustering component to documents already classified positively by the MLKF method making it MLKF+C. This is done to cluster documents which have been classified as cybersecurity-related into groups based on the topic of the document.

Clustering Metrics

We look at the results from the clustering evaluation techniques, silhouette score and davies-bouldin scores. These metrics look at cluster distance and density to generate a score for the quality of clustering which we can use to judge how many clusters should be used. For Silhouette score, a value of 1 indicates well-defined clusters that have well-assigned data points while values closer to 0 and further to -1 indicate overlapping clusters and miss assigned data points. The davies-bouldin score represents the average similarity of each cluster and thus its ideal score would be 0 representing no similarity between clusters meaning clusters represent distinct and unique clusters while a higher score represents non-distinct clusters that are similar and may have overlap. In Table 4.6 we show the scores for all of our datasets and analyze their applicability as a K-value.

We experimented with various vector sizes and cluster amounts from 10 to 2000 and 2 to 10 clusters respectively. In the case of the davies-bouldin score, it shows that low vector sizes perform the best in most cases except the T1 and R1 datasets. In these cases, large vector sizes of 2000 perform best with scores of under 1. The optimal amount of clusters varied based on each dataset but generally had a high amount of clusters in the 9 and 10 range with the T1 dataset having 7 and the R1 dataset having 2.

In the case of the silhouette score, optimal vector sizes are 10 and 2000 with no discernible relationship occurring between feature type and vector size or platform. We can see that in all cases except A1 the number of clusters is set at 2 which speaks to the effect of structured and unstructured data on the Silhouette score. From these results, we have determined that the davies-bouldin score is a more appropriate measure to use to set the number of clusters to use and the vector sizes.

Table 4.6: This table shows the optimal vector size and clusters (K) based on davies-bouldin and the silhouette score for each of our datasets.

	Davies-Bouldin Score			**Silhouette Score**		
Dataset	**Vector Size**	**Clusters**	**Score**	**Vector Size**	**Clusters**	**Score**
F1	10	10	1.988	1000	2	0.162
T1	2000	7	0.561	2000	2	0.565
R1	2000	2	0.860	2000	2	0.507
S1	10	10	1.983	2000	2	0.162
A1	10	9	1.758	10	10	0.139
S2	10	10	2.15	10	2	0.074
M1	10	9	2.081	10	2	0.111
L1	10	10	1.939	10	2	0.222

Clustering Algorithm Selection

We applied silhouette and davies-bouldin scores to our datasets to determine an adequate amount of clusters that could be used based on the distance between clusters and the difference of data points within clusters. The best scores that were achieved on these scores indicate good separation, size, and uniqueness in clusters but good values for these methods do not necessarily map to the best value for information retrieval.

Silhouette scores for all datasets except for A1 had an optimal K-value of 2, showing the greatest separation. In our case since we are aware of the data being made up of various subsites, we are confident that there are more topic categories in each dataset. This is definitely the case for the length-based datasets such as S2, M1, L1, and F1. These datasets are made up of data from various platforms meaning it will have a wider range of topics and therefore a larger amount of clusters. For this same reason, the clustering method is not as useful for our length based datasets and the full dataset. Since these datasets combine data from different platforms we require a larger amount of clusters to retrieve the topic clusters required.

We are able to more effectively cluster datasets based on particular platforms. This is because each platform has particular features such as lexical style, tone, and grammar. This applies itself well to using clustering to identify topic clusters. By clustering based on platforms, we can focus on each platform's particular data better. For the R1 dataset, we do not see much advertising since it is a question and answer site, it contains more posts related to cybersecurity education and certification. For a CTI system, this data is not very relevant and can be removed from the dataset to improve results. These posts do not appear as much on other platforms such as Twitter which has a larger amount of news article advertisements. In the case of the A1 dataset, articles are posted on a website after being edited which is not as common on other platforms and therefore are more trustworthy and do not require the same level of filtering. Each platform has specific characteristics which make it easier to see its particular non-significant data, by mixing data together such as in our length based datasets and the full dataset, it makes it more difficult to distinguish these clusters effectively.

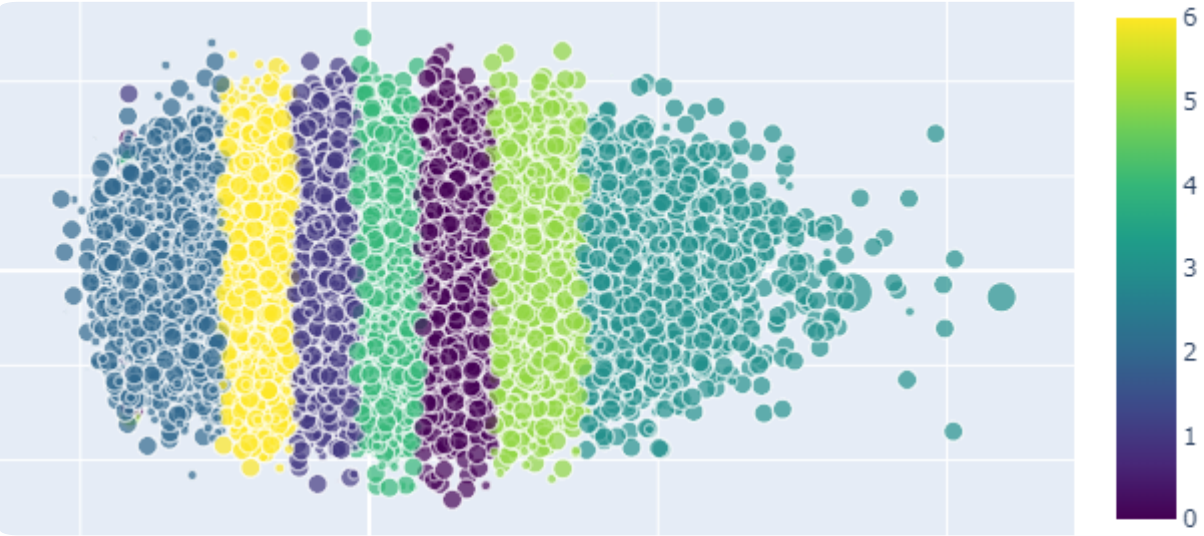


Figure 4.2: This figure shows a two dimensional visualization of our data using the K-means clustering algorithm for topic classification.

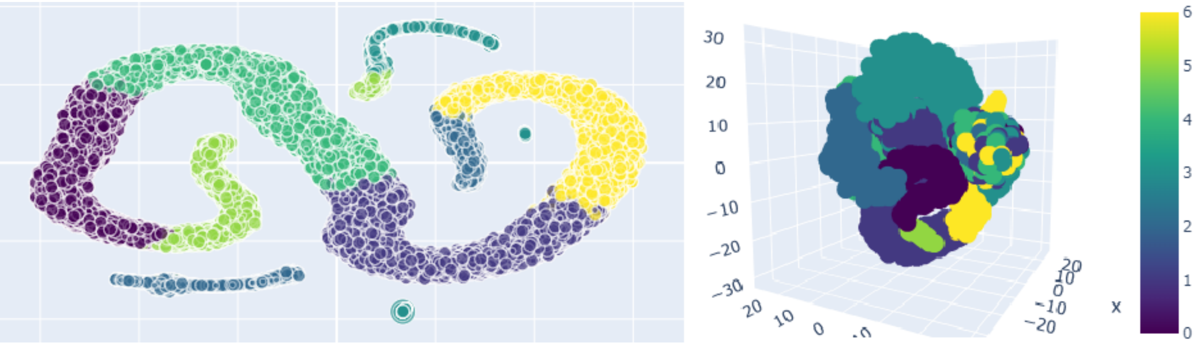


Figure 4.3: This figure shows a two and three dimensional visualization of our data using TSNE and spectral clustering algorithm for topic classification.

Initially, the K-means clustering algorithm was used with the K values obtained through the davies-bouldin score. To analyze the results of clustering using the K-means algorithm on our dataset we visualized our data using the Python library Plotly [119] and used PCA to reduce the dimensionality of our vectors into a two dimensional form. Figure

4.2 shows this visualization, here we can see that projecting our data onto two-dimensional space using PCA produces a blob type form. This does not show us that useful feature selection has been achieved and also that there may be some negatives effects produced from outliers in our data. PCA is effective in a wide variety of contexts and is good as an initial attempt to get an idea of our data. Having said this, it is not the only dimensionality reduction method, so based on the initial attempts we then use t-distributed stochastic neighbor embedding (TSNE) [74] since our data is highly clustered.

In Figure 4.3 we can see the two and three-dimensional representations of our data using the TSNE dimensionality reduction method. This figure shows that TSNE does a better job of representing our dataset, producing better feature representation, and maintaining important relationships between data points. We visualize our data into three dimensions to better understand the relationship between data points. We can see from the three-dimensional representation of our data that classification needs to take into account more features than can be detected in two dimensions. Therefore we can benefit from using a clustering algorithm that is able to distinguish the distance of its closest neighbors from data points that are farther away to achieve better classification. To achieve this we use spectral clustering [92] with affinity set to its nearest_neighoburs. This is a clustering algorithm that uses an affinity matrix which is constructed by computing a graph of nearest neighbors for our data, and in this way is able to distinguish between data points by distance which is beneficial to judging tweet relation and producing better classification results.

Clustering Interpretation

We assessed each cluster by looking at the top 15 terms ranked by TFIDF score in three n-gram categories. This allows us to identify valuable terms in each cluster and topics. Table 4.7 shows the top 15 TFIDF ranked terms for the datasets T1, R1, S1, and A1 using the K-means model. The Davies-Bouldin score suggested 7 clusters for the T1 dataset, using that value we found that there was a separation between clusters and topics. Four

Table 4.7: This table shows the top 15 terms for each K-means generated cluster based on the TFIDF score

Dataset	Cluster	TFIDF Ranked Terms
T1	**0**	cybersecurity, malware, latest, daily, security, phishing, ransomware, cybercrime, via, new, cyber, news, spyware, attack, 2019
	1	cybersecurity, malware, ransomware, security, via, phishing, latest, new, attack, daily, cybercrime, attacks, spyware, cyber, business
	2	cybersecurity, malware, ransomware, latest, security, phishing, via, daily, new, cybercrime, cyber, spyware, attack, attacks, data
	3	cybersecurity, ransomware, malware, latest, daily, via, security, phishing, spyware, attacks, hit, cybercrime, social, viren, trojaner
	4	cybersecurity, latest, malware, daily, cybercrime, security, via, phishing, ransomware, new, spyware, hacking, 2019, news, xss
	5	cybersecurity, latest, daily, security, malware, php, cyber, trojaner, viren, news, technology, shell, phishing, information, ransomware
	6	cybersecurity, latest, malware, daily, security, phishing, ransomware, via, cybercrime, new, cyber, news, spyware, attack, attacks
R1	**0**	security, exam, oscp, time, good, thanks, ceh, experience, course, questions, help, work, looking, take, guys
	1	questions, exam, test, cissp, study, practice, book, read, sybex, time, question, tests, boson, good, took
	2	security, password, network, email, file, data, windows, time, server, account, kali, access, com, user, work
S1	**0**	password, code, key, hash, strong, passwords, random, data, encryption, salt, security, time, attacker, bit, user
	1	code, question, password, answer, security, user, server, strong, thanks, file, site, data, think, google, key
	2	code, file, strong, php, user, script, data, html, server, security, function, name, string, page, content
	3	key, code, certificate, server, client, strong, public, private, tls, ssl, certificates, rsa, security, keys, data
	4	code, org, java, security, springframework, web, user, class, name, spring, public, login, authentication, filter, beans
	5	user, code, password, server, strong, token, security, access, data, client, session, application, users, request, key
	6	strong, security, data, password, information, email, user, access, phone, card, people, code, account, time, question
	7	code, data, strong, system, file, access, user, key, security, password, files, windows, server, software, drive
	8	code, server, network, strong, traffic, port, address, router, access, tcp, vpn, attack, connection, security, firewall
	9	security, strong, code, question, data, answer, good, system, software, access, information, application, web, server, think
A1	**0**	data, security, attack, information, company, breach, attacks, users, bank, malware, accounts, million, customers, hackers, according
	1	malware, win32, file, code, files, malicious, system, com, eset, new, ransomware, user, trojan, banking, exe
	2	facebook, email, information, security, scam, people, account, users, phishing, online, eset, social, scams, malware, data
	3	security, android, password, passwords, app, users, apps, google, new, mobile, devices, phone, apple, device, malware
	4	security, users, google, microsoft, vulnerability, data, facebook, windows, passwords, malware, password, adobe, information, flash,site
	5	security, data, cyber, cybersecurity, information, privacy, report, cybercrime, new, businesses, online, companies, people, attacks, devices
	6	security, data, devices, car, internet, malware, network, device, systems, computer, system, windows, software, information, access
	7	apple, company, percent, uber, inc, yahoo, billion, data, software, new, companies, revenue, last, facebook, google
	8	trump, russia, russian, clinton, election, putin, intelligence, campaign, president, security, officials, state, government, cyber, obama
	9	malware, eset, security, virus, stuxnet, blog, testing, paper, amtso, research, time, article, malicious, information, david

of seven clusters (0, 4, 5, and 6) had news headlines as a prominent topic within them. This type of post is common to see on social media sites with attention-grabbing news headlines often being posted on Twitter as a tactic to bring traffic to a site. It is possible that certain news headlines contain usable data and we can therefore decide which clusters to remove based on these factors.

As previously mentioned the type of non-significant data we wish to identify is often specific to the particular platform. This is shown when assessing terms in the T1 and R1 groups. Reddit has a wide range of subsites that cover a range of topics and are often governed by moderators who keep control of discussion direction. Because of this, we see less advertising on Reddit within our dataset but we see different types of data such as discussions about certification exams and education. This is due to a high amount of students using Reddit for advice about the cybersecurity field either through technical and broader career related questions. These posts are not particularly useful to our system and can be left out.

Ten clusters are used for the S1 dataset which shows various uses of the term code. This is expected as Stack Exchange contains sites such as Stack Overflow in its network which are dedicated to helping users with coding questions. Because of this, it is not surprising that various clusters are identified with a link to programming and have clusters related to specific types of languages and topics. This is shown in the fourth cluster which contains topics related to the Java programming language, and Java security, while cluster three relates to cryptography and associated protocols such as secure socket layer (SSL) and transport layer security (TLS). In these cases we can make decisions based on clusters that may contain unwanted data such as programming code and other unwanted features.

In the case of the A1 dataset, we have less of a requirement to filter advertisement-based data since it is not as common in this form of media. As we can see in Table 4.9 articles are clustered based on article topics such as politics, business, and sport. It is possible to filter out articles based on certain less significant topics such as sport depending on the preferences of the intended system.

Using spectral clustering we achieved unique results with a greater concentration of less significant terms which is closer to desired results. Using the results from clustering our T1 dataset using spectral clustering we show the results in Table 4.9 by ranking the terms for each cluster based on 1-3, 2-3 and 3-3 grams. From this, we can see that various clusters have varying degrees of terms associated with less significant tweets. This is most visible in the third and fifth clusters. In the third cluster, we can see a greater number of terms associated with less significant data points such as "security daily" and

Table 4.8: This table shows examples of non significant cybersecurity relevant tweets taken from the same cluster

	Tweet
1	securityaffairs: Security Affairs newsletter Round 207 “ News of the week. https://t.co/GpR1.. #securityaffairs #malware #hacking
2	The latest The Apex CB Financial Daily! https://paper.li/apexcb/··· #ai #cybersecurity
3	Posts from ThreatsHub Cybersecurity News for 04/01/2019 - https://t.co/stqsoPNrvZ

”latest cyber” which are associated with news headlines. Looking at these terms ranked by trigram we can see that all terms in the third cluster are considered less significant. In contrast to this, the fifth cluster has no terms which would initially be considered of less significance meaning that data clustered into this group would contain higher quality data.

We can therefore use a combination of filtering techniques depending on our application's data requirements, applications that cannot handle any false positives could only use data that is grouped into cluster five while less stringent systems could only restrict data points grouped into cluster three.

Table 4.9: This table shows the top 10 unigram and bigrams and top 5 trigrams for each spectral clustering generated cluster for the T1 dataset using unigrams, bigrams and trigrams based on the TFIDF score

Dataset	Cluster	TFIDF Ranked Terms
T1 - 1-3 gram	**0**	malware, latest, ransomware, thanks, security, phishing, daily, new, daily thanks, cybercrime
	1	malware, latest, thanks, ransomware, phishing, daily, security, cybercrime, daily thanks, new
	2	malware, cybercrime, phishing, security, new, spyware, ransomware, amp, latest, thanks
	3	latest, daily, thanks, security, daily thanks, thanks security, ransomware, malware, cyber, security daily
	4	ransomware, malware, latest, security, thanks, phishing, daily, new, atacks, spyware
	5	malware, checkxs, checkxs ltimg, ltimg, spyware, phishing, amp, cybercrime, ioc, ips
	6	malware, cybercrime, phishing, latest, security, thanks, ransomware, daily, new, spyware
T1 - 2-3 gram	**0**	daily thanks, malware viren, malware viren trojaner, viren trojaner, ransomware atack, thanks security, ransomware atacks, latest daily, cyber security, awarenes month
	1	daily thanks, ransomware atack, malware viren, malware viren trojaner, viren trojaner, ransomware atacks, latest daily, cyber security, hacking cybercrime, phishing scam
	2	daily thanks, pcexpander cybernews, andrews kurth, andrews kurth taps, hunton andrews, hunton andrews kurth, kurth taps, think tank, hacking cybercrime, kurth taps tery
	3	daily thanks, thanks security, security daily, daily thanks security, latest cyber, security daily thanks, cyber security, latest cyber security, latest daily, news thanks
	4	daily thanks, ransomware atack, ransomware atacks, awarenes month, pitney bowes, hit ransomware, ads spyware, social network, mewe nextgen, mewe nextgen social
	5	checkxs ltimg, malware osint, malware osint ioc, md5s ips, osint ioc, ips domains, md5s ips domains, amp lfiamp, amp lfiamp rfi, atacks burpsuite
	6	daily thanks, ransomware atack, hacking cybercrime, cyber security, phishing scam, ransomware atacks, phishing email, reversing cracking, pcexpander cybernews, malware analysis
T1 - 3 gram	**0**	malware viren trojaner, latest daily thanks, daily thanks security, national awarenes month, automation startup tines
	1	malware viren trojaner, latest daily thanks, daily thanks security, qutera web malware, web malware scaner
	2	andrews kurth taps, hunton andrews kurth, kurth taps tery, mcaulife think tank, taps tery mcaulife
	3	daily thanks security, security daily thanks, latest cyber security, cyber security daily, latest daily thanks
	4	mewe nextgen social, network ads spyware, nextgen social network, social network ads, join mewe nextgen
	5	malware osint ioc, md5s ips domains, amp lfiamp rfi, atacks burpsuite xsatack, burp suite amp
	6	malware viren trojaner, aple yanks ios, aps sheltered malware, ios store aps, store aps sheltered

Chapter 5

CTI Practical Application & Implementation

In this research, we are conscious of the practical elements which must be considered when developing a CTI system that aims to be used in a production setting. There are many CTI style systems proposed as research projects which use new algorithms and methods to extract better results or new insights from data. These projects play a role in discovering methods that can be used to improve classification and other processes but in many cases it can take years for techniques to migrate from a research idea into a production system, if they ever do. This is partly due to a lack of consideration for industry applicability in some research, and a greater focus on classification metrics and outcomes.

In this research, we consider the practical elements which influence the adoption of systems in a real-world setting. By considering these factors we are able to outline the recommended scenarios to which it can be applied and highlight benefits and pitfalls. In this section, we consider both technical aspects which must be considered for adoption such as architecture and how that affects classification and processing time, as well as human and business considerations such as user expertise level. By doing this we are able to better outline the scenarios within which this method can be applied in a real-world setting and allow for a smoother implementation of our framework when it is implemented.

5.1 Method

5.1.1 Dataset

In chapter 2 we presented our methods for mining and creating our dataset. This dataset was created using a variety of sources based on social media, news, and discussion platforms. When considering the efficacy of our system there is a great importance in data scope. In this type of system, any potentially valuable information which is published to a platform that is not within the purview of our system will be invisible to us and therefore it cannot be considered or correlated with existing data to assist us in our investigation.

Based on this fact we created a dataset that incorporates four different platforms that have inherently different styles. These platforms were chosen so as to get a representative sample of the different types of data styles that are available and can give us insights into what processes can be applied to other similar platforms through our results. We use Twitter data to provide an example of short and unstructured text which is popular not only on Twitter but through various existing messaging applications. We also use posts from Reddit and Stack Exchange which represent medium length discussion posts that can be both structured and unstructured and contain a large amount of programming code or markup language within them. These sources provide us with examples that can give us insights into how to handle data from traditional style forums which have a similar length and also can contain programming code in the case of malware specific forums. Finally, we incorporate traditional news articles which give us an example of a long structured text which is similar to blogs provided by various security companies. In this way, we have chosen platforms that give us examples of a range of styles that can be applied across a wider scope of platforms we do not currently incorporate in our system.

With this in mind, we have created subsets of our data based on the origin platform of the data and also the character length of the data. By creating subsets of our data based on these features we can consider how these particular subsets can be leveraged to give us enhanced results for specific business scenarios.

5.1.2 Ensemble Method

Classification tasks are traditionally conducted using the dataset as a whole. While this is traditionally the case there are benefits that can be gained by separating data based on inherent features and performing classification on each subset of the data. Although this method can create more work by having to create, train, and test multiple datasets rather than a single one it can have various benefits due to its greater specificity. Different models have different traits and therefore we can apply the appropriate model based on particular subset features and classify each subset individually to get an overall higher classification score or a better processing time. The creation of multiple features based subsets allows us to apply a process called ensemble learning. Ensemble learning is the method by which various classification models can be combined together to achieve a better outcome rather than using a single model.

The benefits that can be gained from this technique depend on how homogeneous the data within the dataset is. In cases where the whole dataset is largely the same, there may be less to gain from separating classification into subgroups. In our case, we know that there is a variety of data with different features within our dataset and therefore we can analyze the effects of using an ensemble method to classify our data. Using this method can have various implications such as increased classification score, processing time, and implementation factors. By segmenting our data in this way and creating separate components we can effectively create different architectures for our system which we can apply to various business scenarios to increase the adoption of our system across various levels of organizations. In Figure 5.1 we show the basic architecture of how we parallelize the classification process through an ensemble learning method.

Traditionally the implementation of CTI systems was considered mainly in large organizations with enough budget and a large enough security team. By using an ensemble method we can achieve a better trade off between effectiveness and practicality which can increase the adoption of these types of systems across organizations that traditionally

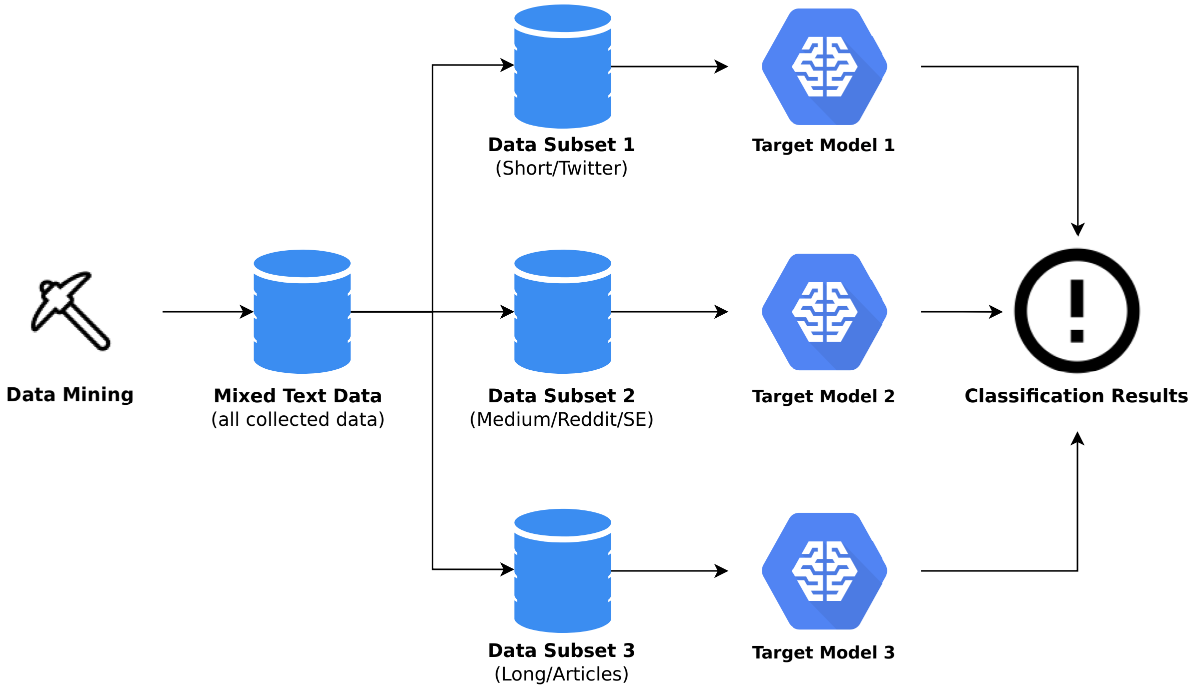


Figure 5.1: This figure shows the basic architecture which can be used in our system when employing an ensemble learning architecture.

were not able to benefit from threat intelligence systems such as small and medium size businesses.

5.2 Results

5.2.1 Ensemble Classification

Table 5.1: This table shows classification scores and processing times for a CTI system implementing the optimal settings based on classification score and speed for both the full dataset and ensemble learning methods.

Dataset	**F1**	**T1**	**S1**	**R1**	**A1**	**S2**	**M1**	**L1**	**Total Precision**	**Total F1-Score**	**Total Time**
Full Dataset (Highest Score)	BERT								0.9877	0.9702	2.12 min
Full Dataset (Fastest)	SGD								0.9304	0.9034	340 ms
Ensemble Learning (Platform : Highest Score)		BERT	BERT	BERT	BERT				0.9742	0.9669	2.2 min
Ensemble Learning (Length : Highest Score)						MLKF	BERT	SGD	0.9826	0.9652	275 ms
Ensemble Learning (Platform : Fastest)		LR	MLKF	LR	LR				0.9448	0.9480	127 ms
Ensemble Learning (Length : Fastest)						LR	SGD	LR	0.9103	0.9114	269 ms

In Table 5.1 we show the results for our data based on using an ensemble method for classification and the time each method takes to process a sample of 100 data entries. This table shows the optimal model or combination of models to use based on which dataset features were present. This allows us to compare different scenarios based on the various implementations. We show the results of a traditional classification method also where we use the full dataset for training and there is no feature based segmentation used. In this case, we achieve the highest f1-Score, albeit by a relatively low margin of 0.003. Although it is able to achieve the highest score the total time taken to process data is the second lowest of all scenarios with 2.12 minutes. When we look at the optimal model for the full dataset based on speed factors we get a faster total time but a significantly lower f1-Score. These factors lead us in the direction that using the full dataset is not the ideal implementation for our use case when considering both classification score and the processing time needed for a real-time system.

If we look at the ensemble learning methods that segment our data based on platforms, in the case which achieves the highest score we also have the highest total processing time which make the scenario impractical. This shows that the BERT model and thus a transfer learning method is powerful in terms of classification, but it is not as practical for real-time systems due to slow speed. Although this is generally the case, with enough computing power this can be alleviated depending on the speed at which entries are being input into the system, and a high enough budget to implement the solution.

If we look at the ensemble learning method which was subdivided by length and achieved the fastest total time we see that although it is faster than its high score pair it is not faster by a large amount. While when we consider the F1-Score it is significantly lower with 0.9114. The two methods which give the best trade off between speed and classification score in our use case are both ensemble learning-based methods with the length subset which received the highest score and the platform subset which had the fastest processing time. When we analyze the length subset which received the highest score. Its f1-Score is third overall but is only 0.005 below first place, meaning it has a very good classification score. Alongside this, it has a much faster overall processing time compared to the model which has the highest score with 275ms.

In the case of the platform subset which had the fastest processing time, we see the fastest total processing time of all scenarios with 127ms with a comparable f1-Score of 0.9480. Although the f1-score is slightly lower than the closest competitor, it is 46.18% faster at 127ms.

These results show that while using the traditional method of a single dataset technically achieved the highest classification results this aspect was so closely matched by the other scenarios that it is not significant. This coupled with the fact that it is a much slower technique in terms of processing time makes it less practical for real-world implementations compared to an ensemble learning method. Even though we show from our results that an ensemble learning method is preferred there is a difference between methods depending on dataset features used and the models chosen.

The two methods which are most appropriate for a CTI system are made up of multiple models. We are able to achieve these results by applying the most appropriate model to a subset of data based on the two metrics we have chosen of f1-Score and total processing time. Our MLKF method appears in both of the recommended scenarios. The MLKF method was able to achieve the best score on the S2 dataset and the best processing time on the S1 dataset. This shows that our method can be an effective part of a CTI system to achieve better results when classifying and processing cybersecurity data.

5.2.2 Processing Time

In the previous section, we looked at the processing time based on various database considerations. When implementing this type of system in a real-world scenario there are various factors that affect this and have to be considered. In this section, we go into more detail on other areas that affect the processing time of our method such as architecture and how much time is needed for the pre-processing and classification stages.

Table 5.2 presents the time required to process a subset of our data for each particular classification model. These subsets are comprised of 100 entries with each entry having a length equal to the average post length for that dataset, this gives us the average processing times for each model. This table shows the total processing time for both CPU and GPU on the pre-processing method as well as on the full classification method. An increase in GPU use in machine learning and artificial intelligence research has allowed neural network models to become more applicable in industry, being able to train models using GPUs have made neural network-based models and deep learning more feasible for various applications [122]. Because of this and GPUs usefulness in other areas, GPUs have become more expensive, harder to acquire, and they are not present in all computer systems. CPUs on the other hand are used in all computer systems. Because of this a company or individual without access to a GPU will benefit from a solution that is viable on a CPU. We test on both CPU and GPU based systems, using an Intel Xeon 2.30 GHz CPU with 13.3 Gb of memory, and an Nvidia Tesla P4 with the same amount of memory.

Table 5.2: This table shows the total processing times in milliseconds for a processing only version and full version of our method across all datasets for both CPU and GPU processors for all the models we tested.

Dataset	Processor	Mode	MLKF	BERT	LR	SGD	NB
F1	CPU	Processing	174 ms	1140 ms	138 ms	144 ms	137 ms
		Full	790 ms	127200 ms	487 ms	358 ms	374 ms
	GPU	Processing	141 ms	1040 ms	126 ms	120 ms	125 ms
		Full	647 ms	2330 ms	474 ms	347 ms	340 ms
T1	CPU	Processing	12.5 ms	65.3 ms	9.44 ms	10.9 ms	8.99 ms
		Full	25.4 ms	127200 ms	23.1 ms	20.4 ms	17 ms
	GPU	Processing	9.36 ms	62 ms	8.5 ms	8.09 ms	8.38 ms
		Full	19.4 ms	1290 ms	18.5 ms	13.9 ms	14.4 ms
R1	CPU	Processing	65.3 ms	368 ms	45.4 ms	48.6 ms	45.4 ms
		Full	240 ms	132000 ms	71.7 ms	73.5 ms	66.7 ms
	GPU	Processing	55.1 ms	343 ms	41.1 ms	40.2 ms	40.8 ms
		Full	207 ms	1650 ms	63.3 ms	55.3 ms	58.9 ms
S1	CPU	Processing	13.4 ms	68.5 ms	9.82 ms	10.8 ms	10.3 ms
		Full	172 ms	129000 ms	267 ms	170 ms	167 ms
	GPU	Processing	11.1 ms	60.7 ms	8.64 ms	8.46 ms	8.49 ms
		Full	127 ms	1290 ms	233 ms	142 ms	161 ms
A1	CPU	Processing	218 ms	1370 ms	168 ms	10.6 ms	167 ms
		Full	635 ms	83400 ms	291 ms	38.4 ms	269 ms
	GPU	Processing	186 ms	1270 ms	151 ms	144 ms	153 ms
		Full	558 ms	2290 ms	256 ms	236 ms	246 ms
S2	CPU	Processing	12.7 ms	51.7 ms	9.39 ms	11.1 ms	9.84 ms
		Full	43.8 ms	39100 ms	35.8 ms	28 ms	25.7 ms
	GPU	Processing	9.73 ms	46.5 ms	8.32 ms	8.15 ms	8.6 ms
		Full	35.2 ms	425 ms	27.3 ms	19.6 ms	20.9 ms
M1	CPU	Processing	39.8 ms	219 ms	28.6 ms	31 ms	28.1 ms
		Full	226 ms	121200 ms	215 ms	145 ms	142 ms
	GPU	Processing	31.9 ms	202 ms	25.2 ms	23.8 ms	25.6 ms
		Full	180 ms	1500 ms	189 ms	133 ms	116 ms
L1	CPU	Processing	159 ms	1170 ms	137 ms	143 ms	137 ms
		Full	487 ms	129600 ms	355 ms	318 ms	298 ms
	GPU	Processing	139 ms	1060 ms	122 ms	117 ms	125 ms
		Full	421 ms	2300 ms	343 ms	269 ms	275 ms

To calculate the processing time we used the Python timeit library.

We tested two parts of our method separately. The pre-processing phase which deals with manipulation and cleaning of data into a form that can be used by the classification model was tested separately from the classification itself to see the effect data length and data type have on processing time. We also test the full method from data processing through to classification which shows how much time the whole method takes.

In Table 5.2 we can see clear patterns arise which show how various models are affected. As may be expected for machine learning models GPUs are faster than CPUs in both the pre-processing and classification-based tasks with obvious variability depending on the specific dataset used. Documents that were longer and hence had a larger dataset took longer to process. This can be seen through the A1 dataset processing times which took the longest to process while the datasets which were made up of short documents such as S2 and T1 have lower times. The S1 dataset has low processing times similar to those of the T1 and S2 datasets even though it has a significantly higher average document length. As we know the S1 dataset contains posts from the Stack Exchange network websites which often contain a large amount of programming code and markup language. Because of this, we can get an effect where more work is being done in the pre-processing stage and leaving smaller documents to be processed during the classification phase. While the documents may start off with a longer average length after the pre-processing phase they are reduced to a much smaller size. Because of this, we can see a difference in that a dataset with a longer average length has a lower classification time. Similarly, this is the case with the R1 dataset, although it does not contain as much removed data as the S1 dataset it still has enough to reduce the classification time. The processing time for the full method shows similar results on both processors, with datasets such as A1, L1, and F1 generally taking longer due to their longer length.

The BERT classification step takes a large amount of time along with the naive bayes model which has a significant increase in classification time showing a more time-intensive classification process when compared to linear regression and stochastic gradient descent.

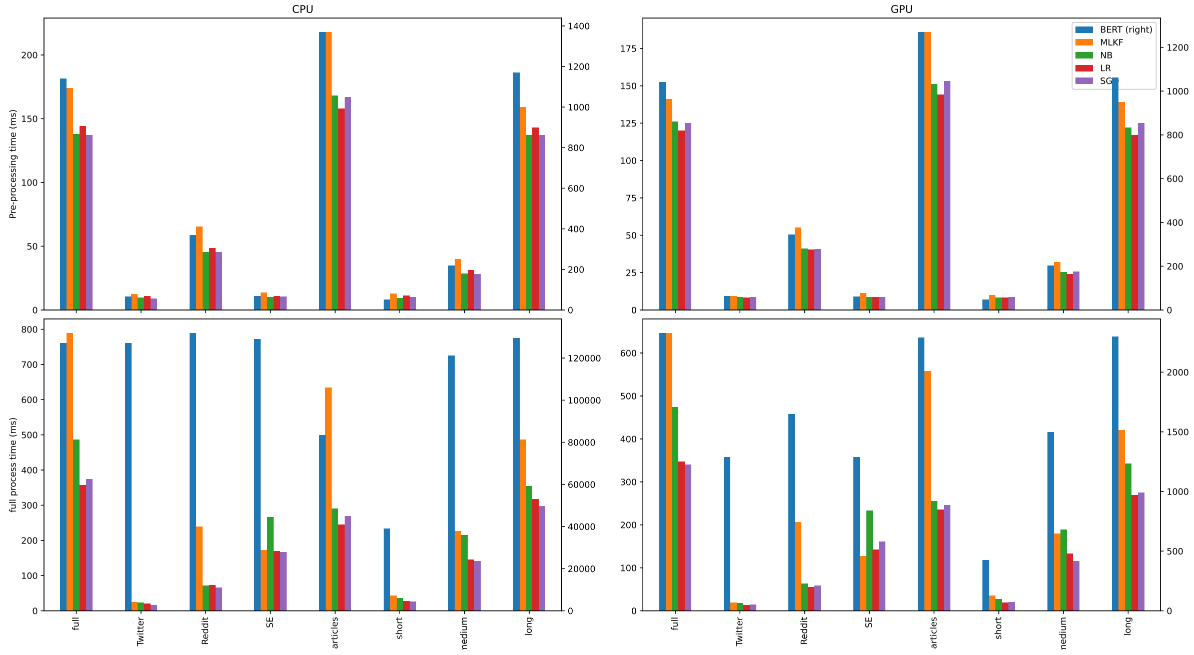


Figure 5.2: This figure shows a barplot of the processing times for each dataset based on processor type, as well as pre-processing and the full method measured in milliseconds.

For the BERT model, the classification step takes significantly longer than pre-processing, with BERT also being slower than all other models for data pre-processing. As we have previously mentioned this produces problems for its application in CTI systems which uses real-time data from various sources. This can cause issues such as a system crashes since it can not deal with the amount of data being processed. Utilizing multiple GPUs will alleviate this and would even be expected in larger implementations. However, this can be impractical for most companies to implement due to its high implementation cost, and the option to use the models which have comparable classification scores with better processing times are a better choice.

A characteristic of mining data from multiple sources is that each platform has varied ”post times”. For example, a platform like Twitter has 6000 tweets per second [121] while news sites have a much slower rate of anywhere between 200 and 500 articles per day [84]. Based on our test using the BERT method with our particular system, it would take 1.174

minutes per data point which is too slow for a real-time system using Twitter. Therefore using an ensemble learning method we can select the most appropriate model for each scenario to achieve the best trade off for a particular implementation. For a smaller organization or one that does not have the resources or infrastructure to implement an ensemble solution, the MLKF and baseline methods can be used to implement a solution.

Finally, the MLKF coding implementations are not production based systems and therefore have room to be optimized to gain faster processing times. The baseline models were implemented using the machine learning Python library Scikit learn, which has been used for production systems and therefore has been more rigorously tested than our method. Even though our current times are close to the baseline learning times an various datasets with more optimization our method could surpass current times.

5.2.3 Scalability, Budget & Other Considerations

Apart from the main considerations of classification score and processing time, there are a number of other aspects that must be taken into account when implementing this type of system. In the case of a large company that has many skilled employees, a large budget, and adequate infrastructure, the implementation of this type of system would not be something unusual. In the past, there has often been an image that only large companies can effectively implement and benefit from a CTI system. This could possibly be the case because the existing solutions for implementing a CTI system were catered towards larger companies and did not provide a pathway for smaller companies that had lower expertise and experience preventing adoption. In this section we address some of these extraneous factors which need to be considered in the various use cases for a CTI system which will allow our method to be applicable in different real-world implementations.

Firstly, as previously shown the ensemble method provides a good trade off with both the length subset which received the highest score, and the platform subset which had the fastest processing time being acceptable choices for use in a CTI system that has to consider both infrastructure and budget. This would be the ideal middle ground and would suit itself best to small to medium sized companies which want the best value.

In the case where one particular measurement needs to be optimized such as classification score the full dataset scenario using a model such as BERT can be utilized but is often not practical depending on outside factors. A solution that uses BERT within the parameters which we have shown has proven to be limited by processing time constraints. This can be alleviated by using more processing power and infrastructure to make it more feasible. This is a solution that would most likely only be available to certain organization that possesses the budget and expertise available to implement it. Even in this case depending on the speed at which data points arrive and need to be classified (which is generally a high speed in real-time systems such as this one) it can be overwhelming and not make it practical. Having said this using an ensemble method based on platform segregation we can separate our datasets into subsets based on platforms to achieve results which are more usable. Inherently, different platforms receive data at different rates, with traditional news articles being much slower than real-time systems like Twitter. This can allow for a more processor-intensive model to be applied to a platform that does not receive data as frequently and make slower models more feasible.

Another factor which must be considered is the use of a single dataset, platform-based ensemble, and length based ensemble. The way in which these model architectures affect the infrastructure and maintenance needed in a production system must be considered. Although we have shown the ensemble-based method is more applicable and recommended for our method, its real-world implementation brings with it other pros and cons. In the case of implementing this method using a single dataset rather than subsets of datasets may allow for a single higher-powered server, virtual machine or container to be used for a single dataset. Because of it being a single dataset there is a single point of failure if there is a problem with the implementation. Although this could be remedied with backup servers there would still be some transition period and higher maintenance. Using a single dataset also has implications for the time it will take to train new models being longer due to the larger amount of data.

In the case of the ensemble method rather than have a single server or container for all datasets it would be more appropriate to have multiple servers, containers, or virtual machines to host individual datasets and models. This has the obvious effect of causing more work because there are more models to be trained and infrastructure to be maintained, although this can have various benefits when it comes to the smooth running of the service. By separating each model into different containers we have the benefit of training and maintaining each container separately rather than having to do it all at once which is the case with a single dataset model. Also if there are any issues with a model crashing or another fault, the system will continue to function minus that particular component rather than lose access.

Using our method has an implication that is greatly affected by the data segmentation type chosen. Choosing to segment data based on platform or length can have an impact on the number of models and infrastructure which have to be created. Using a length-based data segmentation method the number of models and containers to house them will be fixed based on the length groupings which have been decided originally. The size of these models will grow as more sources are added to the system meaning we have a situation where we have a fixed number of models that can become larger over time as the amount of input sources grow. On the other hand by segmenting our data based on the platform we need to create a new model and container for each platform. In a large implementation, this can be an issue since there are a large number of platforms, and a CTI system benefits from having a wide scope of platforms from which it takes its data from. This causes the obvious issue in that there may be a large number of servers or containers required to house each model and dataset which will cause a large number of overhead for employees due to maintenance and training. There is a third option of grouping platforms based on post similarity this requires further investigation into how similar the posts found on two platforms actually are, and if combining them achieves an appropriate classification score. These factors need to be considered in conjunction with the number of employees available to create and maintain the system as well as the level of expertise of the employees to troubleshoot problems. It can also have budget implications if each model is being created on new servers. These various factors need to be considered during the real-world implementation of this system to achieve a successful implementation that can be used effectively over time.

Chapter 6

Discussion

In this section, we discuss the considerations of our methods including the factors which limit their efficiency and usefulness. This is done by going through our design, methodology as well as possible alternatives. By highlighting these points of our methods we allow users to consider these aspects in the implementation of our methods and the analysis of the output of an implemented version of our system.

6.1 MLKF+C Filter Considerations

The MLKF+C method was developed as a way to enhance the data quality for systems that have the intended use case of a CTI system. This method therefore has inherent strengths and weaknesses which favor this use case which may not be considered positive for other use cases.

Our system obtains data based on the MLKF+C [105] method which generates a stream of cybersecurity data by implementing context-based keyword filtering using word2vec, and clustering to limit the amount of advertising and marketing posts in the data stream. This method produces good results for our use case and has effectively produced a dataset containing relevant cybersecurity data. Although this output does not contain significant out of domain data it does contain aspects that users should be conscious of. Although non-significant data points such as marketing and advertising data are limited by this method they can still be found. It was expected that this type of data would still find its way into the dataset but by limiting its integration we ensure it has minimal to no effect on the system processes and output. Because of the small amount of data which is incorporated in this way it does not have a significant affect on our system. The MLKF method also incorporates a manually kept keyword filter, which works as the basis for the initial seed words that are retrieved by the system. The manual configuration employed provides more user control over the system but also requires more user attention which can be prone to user error. An example of this came in July 2020 when political ads were released using the term ”trojan horse” being used outside of a cybersecurity context. This term was not included in our double meaning word filter and therefore included political tweets which were note related to cybersecurity. Due to filter configuration, this data was incorporated into the dataset and quickly took up a large number of entries because of widespread coverage of the event being much more mainstream compared to cybersecurity events. By configuring our filters further to process the term ”trojan horse” through the word2vec model this was fixed and unrelated data for this event no longer affected the system. This shows that there is a need for customization and configuration with this method concerning updating the manual keyword filter and double meaning word filter to fit particular use cases. Using a manual keyword filter method was an intentional choice to give the user more control over output but using this method does increase the level of maintenance required.

In a similar way, the scope of terms used can have a significant effect on the number of false-positives and false-negatives which are incorporated into the dataset. Using a wide keyword filter with a large number of terms including ambiguous terms will ensure a wider scope of data will be incorporated, but this will come with a greater amount of false-positives. This option could be used if the application of the system wanted to incorporate more edge cases or outliers such as obscure posts. Alternatively, if we use a narrow scoped filter we can expect much fewer false-positives but a much narrower scope of data. That is to say, the data which is retrieved will be more reliable but it will be less likely to contain obscure posts or edge cases. The scope of the filter for these cases is affected significantly by the stemming process also. If some form of stemming is used it will effectively narrow the scope of the filter further by reducing entries to their common stem. These factors must be considered in downstream applications based on the type of data required and the amount of acceptable false-positive and false-negatives. The filter should be set according to the type of data that is expected to be retrieved and used within the system.

MLKF+C can produce more false-positives with medium-length unstructured texts. This was shown in the results for the MLKF method where although our AWL has the highest improvement rate for medium length posts it generally produces the lowest classification scores compared to the other datasets in these types of documents. This is mainly attributed to two factors. Firstly, lexical similarity of the post is important in determining its relevance. Because of the wide range of topics found on social media and related news platforms we get a wide range of discussions. It is easy to see how a discussion about animals would have less lexical similarity with a discussion about cybersecurity. Therefore as the context between discussions becomes closer so does the potential for false-positives. Many technical discussions in the area of computer science, computer engineering, and other areas are very similar in context to that of cybersecurity and therefore when dealing with these documents it is more likely to get false-positives. These documents are more likely to appear on certain platforms and have certain features which we can use to limit this impact. For example, we can use a different model with a narrower scope on documents that are from a specific platform while using a wider scope on documents from another. Secondly, the length of the document has an obvious effect on accurate classification. For medium and long length documents there is a greater chance that there will be a match between a term from the AWL and a term within the document simply because of its length. Similarly, we can consider the length of the AWL list used to regulate classification on these documents. One example of an issue that can arise from this is long texts such as articles that cover various topics. There are articles that cover a range of topics that are tangential to the main topic within the article. This can mean we incorporate a document which is technically relevant but may have low significance. This needs to be considered also when dealing with articles and other long-length data.

6.2 Emerging Threat Term Incorporation

We also have to consider the integration of new threat and attack names into our filters to keep the results up to date and relevant. Although in our initial framework we show that TF and TFIDF can be used to distinguish relevant words that can be considered as candidates to be entered as new threat terms, the ultimate decision of adding these words is still one that must be done manually. Because of the manual nature of term selection for our main filter, there is a time period between the emergence of a new attack and the addition of that term to the filter. This time period can affect the level of incorporation that this type of data is integrated into the dataset. Having said this, when a new attack appears it is often qualified with its attack type in posts. That is to say, if a new ransomware threat appears with an obscure name it will initially be found in posts which have the term ransomware in the same post. Over time if the threat becomes more popular it is more likely that the actual name is used without the attack type as it becomes more known in the community. Because of this, there is some overlap where new threat terms are incorporated into the dataset even though the particular name of the attack has not been added to the filter. This allows administrators some time to triage and update the filters. Threat and attack names can change or appear rapidly and often are not terms that previously existed, because of this there is still no adequate method to reliably determine these terms automatically without user intervention. Therefore users need the power to select these terms and incorporate them into the system as they appear, even though this does come with the detractor of requiring more manual work and user consideration.

6.3 Topic Popularity

When considering the downstream issues which may arise from the implementation of our system we can see how the popularity of a particular topic that moves to the wider audience can affect the analysis of our output. In the case of Twitter, even though it has a large cybersecurity community who uses it to discuss various topics, cybersecurity discussion only makes up a small amount of the overall conversation on Twitter. Because of this when certain cybersecurity topics become mainstream, large amounts of this data is recorded for those particular threats. For example, on August 1st, 2020 there was widespread news of a mobile application being spyware and its potential banning in certain countries. This caused a large spike in social media and related sources referencing and discussing this topic. This meant our system recorded a large spike for ”spyware”. Our tool accurately reflects the amount of interest for the topic based on its widespread popularity, but when subjects become mainstream they garner more attention and can dominate other threats which should be a consideration in downstream applications. One effect of this is that the subject can take over multiple places in a ranking system pushing other information out. This is due to the amount of data related to that subject and also the inevitable amount of retweets which significantly increases occurrences of the same or similar posts. We aim to accurately represent the significance of topics when they reach wider audiences, but these events do not always accurately reflect the threat level. When a particular topic becomes politicized or popular it is not always an accurate reflection of the amount of risk it poses to systems. Therefore the downstream application needs to take this into account if they want to consider the actual risk level of a threat compared to the popularity of that threat. Having said this there are examples such as Heartbleed and wannacry which garnered a large amount of mainstream attention because of their high risk level and the damage caused, but this is not always the case. Therefore depending on the application, this consideration needs to be kept in mind so that particular data does not overpower other entries. One way to handle this is to consider using a normalization method to adequately represent that subject in a ranked form. When doing this it should be considered to limit the number of places which it can take up in the rankings. This is done because of the large numbers of the same data present in the dataset which can cause issues in downstream applications such as making the investigation more cumbersome. This can also be remedied by consolidating very similar and identical tweets into single representative Tweets.

6.4 Dataset Creation

In this research, we use our own created datasets to test and compare our methods. The emerging nature of CTI research in the application of ML and NLP technologies to cybersecurity text classification has meant that there is not a specific gold standard dataset for this area, and a lack of large datasets in general. Because of this, we have created our own datasets for training and testing of our models. These datasets were created by retrieving data from social networking sites and other related media sites. These documents were automatically annotated based on the origin of the document. We selected specific sources that contained features such as user-defined tags and hashtags which represent user based annotation. We also used topic-based subsites, rooms, and discussion boards which were limited to cybersecurity discussions which used moderators. By limiting our data retrieval to these sources for our positive data we automatically annotated our dataset based on the inherent features of these sources. This method is beneficial because it allows us to have a large dataset that is annotated into binary classes on a scale which would be otherwise unpractical with a manual annotation method. However, the question can arise with automatic annotation of a dataset as to the validity of the test set and the amount of noise within the dataset. These potential biases in the dataset must be considered through erroneously labeled data points. An example, can be a post that is tagged by a user as Python related which is not related to Python, or the inverse, a post which is related to Python but has not been tagged as such. A perfect dataset of significant size is not practical and therefore some level of erroneous labeling must be accepted with this annotation method. The important point in this scenario is to be able to show that the amount of erroneous labeling is minimal and that it does not affect the validity of the results which are achieved from using the dataset.

To do this we use a random sample that equally represents the various platforms in our dataset. This sample set is made up of 500 entries which we manually annotated to find how many entries were correctly and incorrectly labeled. From our manual annotation, we found that the annotation was effective with 98% of entries being labeled correctly. This means we have a p-value of 0.9782 which allows us to judge with 99% confidence that through normal approximation we have labeled our dataset correctly. This gives us a lower bound of 0.9614 and an upper bound of 0.9949. In this way, we can see that the level of noise which is represented in our sample set and can be projected onto the greater dataset is of an acceptable level and does not affect the results which were gained using this dataset.

6.5 Class Distribution & Data Type

Within the dataset, we have two classes which are cybersecurity-related data and noncybersecurity related. The class distribution refers to the amount of each of these classes which is present in our various datasets. The level of class distribution within the testing set can affect the classification results and therefore has to be considered during testing. The class distribution should take into account the class distribution which will be present in the particular use case of the system. In our case, we have a balanced class distribution for the majority of our created datasets. This class distribution is not inherent in the platforms which we use as a whole. Taking Twitter as an example, cybersecurity posts make up a small fraction of the total discussion and therefore if a system is created which is taking data directly from Twitter without any previous curation the testing set should reflect this class distribution accordingly. In our case, we are retrieving our data from a pre-processed or curated state which skews the data to contain more positive examples. Our use case takes the stance that rather than need to initially retrieve all data from any particular source we can pre-filter it based on the particular subsite, hashtag, or user tag which means we will have a much larger amount of positive examples for our class distribution. In the case of platforms that do not have a feature such as a tag, or moderated subsite the class distribution used in the testing set for those scenarios should reflect that use case.

Our system aims to incorporate a wide scope of data and cover all appropriate cybersecurity topics. By doing this we find that we also incorporate general cybersecurity data, such as defense "tips" and "advice". This type of data is found less frequently than posts about threats and therefore does not take up a prominent position in our output, but can on occasion have an effect. This type of data is relevant especially if it is about an attack. It can be useful for users if a fix has been found to a particular threat or if an important patch has been released. This type of data can often have a positive polarity due to the context it is used in having a positive implication. Both data about active threats and fixes are valuable in our dataset. Both types of data can benefit users and can be used to judge the severity of a threat. For example, if a fix is found for a particular threat then its risk level has been dropped significantly, and therefore its impact and the steps an analyst would take to protect themselves from it would be altered. These aspects must be considered when dealing with posts which have a positive sentiment within our system.

Chapter 7

Conclusion

In this research, we have presented three principal contributions. Firstly, we have presented a framework for a real-time cybersecurity situational awareness system that can assist analysts and other interested parties in achieving greater insight into the security landscape. This is done by gaining better knowledge of the current sentiment towards organizations and companies within a specific industry. Negative sentiment data which is tied with certain attack terms and company names in posts can be taken as a precursor to a cyber attack. Based on this concept we develop the various components required to create a CTI system. To implement this framework we bring together components such as a data filtering method that retrieves cybersecurity-related data from specific security accounts on the Twitter platform. Once we have retrieved our data we can then process and categorize our data including grouping it based on polarity. Finally, we implement data visualization for our method that is able to show the rate of posts that include a negative sentiment in conjunction with attack terms. When we correlate this information with the mention of a specific organization such as a company, government, or person, this can provide us with valuable insights that can be applied by analysts to lower the risk level of an attacker harming their network. We also found that this system was able to provide earlier detection when compared to traditional cyber security news site methods with it detecting threats faster in 87% of our tested cases.

Secondly, from our experience working on our framework for a cyber threat intelligence system, we were able to gain valuable insights into which components proved to be more important when considering the output of the system and we were able to see how crucial the initial data acquisition phase of the system is. This importance of this phase is present throughout the computer industry and is represented by the phrase "garbage in, garbage out". This signifies that the effectiveness and usefulness of the whole system depends on the quality of the data that is inputted into it. With this in mind in our second contribution, we developed an algorithm that greatly expands upon the initial methods used to enhance the quality of data that can be inputted into our proposed system. This method is called multi-Layer keyword filtering and clustering (MLKF+C) and it uses a series of word filters including a dynamically generated layer to disambiguate data points and to classify the relevance of a post. Using this method we achieve an f1-Score of 0.99 on various subsets of our data and do especially well with short unstructured text data such as Twitter data. This can be attributed in part to the addition of our associated word list which gives up to a 7.8% increase in f1-Score. We subsequently perform clustering on those posts to ensure that only significant data is integrated into the data stream. Through this we have created a lightweight algorithm that can be used for data classification within a cyber threat intelligence system. This algorithm uses an expanded version of keyword filtering, a method that is common in academic research in cyber threat intelligence systems and in production systems and therefore provides an alternative that can be implemented to enhance the quality of data used in those domains.

Lastly, we analyze and look at the effective implementation and adoption of our proposed system. Although having a working system that provides accurate results and good insights is the goal of any system, these are not the only factors that contribute to their adoption. There are various financial, business, and human factors that contribute to the adoption of a particular system apart from its effectiveness, these include but are not limited to, cost, expertise, infrastructure, and more. Therefore we examine how these factors affect our system in particular and how these factors can affect the adoption of our system. One of those considerations is the segregation of data and how data can be processed differently based on inherent features and the effect this has on architecture. By segmenting data based on specific features we can apply dedicated models to each dataset and gain an improvement in speed and classification. This segmentation also has an effect on data storage maintenance, training, and implementation which influences the scenario in which it is applied. These factors include processing time and architecture factors which we investigate to show the best trade-off in different implementations of our system. By using an ensemble method we were able to achieve a high F1-Score of 0.9652 with a processing time of 275ms achieving a good trade off between speed and classification. This particular implementation is a good trade off between speed and classification but other implementations can be used to either maximize classification or maximize efficiency. In this way we can provide an adequate solution across various scenarios including small and medium businesses which traditionally have a lower adoption rate for cyber threat intelligence systems.

Together, this research provides a pathway for the effective implementation of a cyber threat intelligence system that is able to effectively filter and integrate open-source data and how to apply our proposed method with organization considerations in mind to stand up a compelling threat intelligence program.

Bibliography

[1] Selenium documentation, 2013.

[2] Cisco 2018 annual cybersecurity report. Technical report, Cisco Systems, 2018.

[3] The impact of security alert overload. Technical report, CriticalStart, 2019.

[4] Lillian Ablon and Andy Bogart. *Zero days, thousands of nights: The life and times of zero-day vulnerabilities and their exploits*. Rand Corporation, 2017.

[5] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad hoc networks*, 2(1):1–22, 2004.

[6] Charu C Aggarwal, S Yu Philip, Jiawei Han, and Jianyong Wang. A framework for clustering evolving data streams. In *Proceedings of the 2003 VLDB conference*, pages 81–92. Elsevier, 2003.

[7] Muna Al-Hawawreh, Frank den Hartog, and Elena Sitnikova. Targeted ransomware: A new cyber threat to edge system of brownfield industrial internet of things. *IEEE Internet of Things Journal*, 6(4):7137–7151, 2019.

[8] Khalid Al-Rowaily, Muhammad Abulaish, Nur Al-Hasan Haldar, and Majed Al-Rubaian. Bisal–a bilingual sentiment analysis lexicon to analyze dark web forums for cyber security. *Digital Investigation*, 14:53–62, 2015.

[9] Rodney Alexander et al. Using linear regression analysis and defense in depth to protect networks during the global corona pandemic. *Journal of Information Security*, 11(04):261, 2020.

[10] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.

[11] Mohammed Almukaynizi, Ericsson Marin, Eric Nunes, Paulo Shakarian, Gerardo I Simari, Dipsy Kapoor, and Timothy Siedlecki. Darkmention: A deployed system to predict enterprise-targeted external cyberattacks. In *Proceedings of the 2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 31–36. IEEE, 2018.

[12] Jorge Alvarez and Peter Nuthall. Adoption of computer based information systems: The case of dairy farmers in canterbury, nz, and florida, uruguay. *Computers and Electronics in Agriculture*, 50(1):48–60, 2006.

[13] Fernando Alves, Aurélien Bettini, Pedro M Ferreira, and Alysson Bessani. Processing tweets for cybersecurity threat awareness. *arXiv preprint arXiv:1904.02072*, 2019.

[14] Nor Badrul Anuar, Hasimi Sallehudin, Abdullah Gani, and Omar Zakaria. Identifying false alarm for network intrusion detection system using hybrid data mining and decision tree. *Malaysian journal of computer science*, 21(2):101–115, 2008.

[15] Çağri Burak Aslan, Shujun Li, Fatih V ÇELEBI, and Hao Tian. The world of defacers: Looking through the lens of their activities on twitter. *IEEE Access*, 8:204132–204143, 2020.

[16] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 830–839, 2020.

[17] Vahid Behzadan, Carlos Aguirre, Avishek Bose, and William Hsu. Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream. In *Proceedings of the 2018 IEEE International Conference on Big Data, Big Data 2018*, pages 5002–5007. IEEE, 2018.

[18] Steven M Bellovin. Security problems in the tcp/ip protocol suite. *ACM SIGCOMM Computer Communication Review*, 19(2):32–48, 1989.

[19] Angel Belzunegui-Eraso and Amaya Erro-Garcés. Teleworking in the context of the covid-19 crisis. *Sustainability*, 12(9):3662, 2020.

[20] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[21] Sandeep Bhatt, Pratyusa K Manadhata, and Loai Zomlot. The operational role of security information and event management systems. *IEEE security & Privacy*, 12(5):35–41, 2014.

[22] DAVID BISSON. Popular web browser 's hidden ability threatens 500m google play users, 2019.

[23] Frans Botes, Louise Leenen, and Retha De La Harpe. Ant colony induced decision trees for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security*, pages 53–62. ACPI, 2017.

[24] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[25] Cornelia Caragea, Adrian Silvescu, and Andrea H Tapia. Identifying informative messages in disaster events using convolutional neural networks. In *International conference on information systems for crisis response and management*, pages 137–147, 2016.

[26] V Cerf. An assessment of arpanet protocols. *Network Systems and Software (Infotech State of the Art Report 24), Maidenhead*, pages 462–4, 1975.

[27] D Brent Chapman, Elizabeth D Zwicky, and Deborah Russell. *Building internet firewalls*. O'Reilly & Associates, Inc., 1995.

[28] Haipeng Chen, Rui Liu, Noseong Park, and VS Subrahmanian. Using twitter to predict when vulnerabilities will be exploited. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3143–3152, 2019.

[29] Jay Chen. The state of exploit development: 80cves. Technical report, Unit 42 Palo Alto Networks, 2020.

[30] Federico Concone, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. Twitter analysis for real-time malware discovery. In *Proceedings of the 2017 AEIT International Annual Conference*, pages 1–6. IEEE, 2017.

[31] Chris Crowley and John Pescatore. Common and best practices for security operations centers: Results of the 2019 soc survey. *SANS, available at https://www. sans. org/media/analyst-program/common-practices-security-operations-centersresults-2019-soc-survey-39060. pdf (accessed 25th July, 2020)*, 2019.

[32] RL DAVID. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings of DARPA Information Survivability Conference and Exposition, 2000*, 2000.

[33] Isuf Deliu, Carl Leichter, and Katrin Franke. Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent dirichlet allocation. In *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, pages 5008–5013. IEEE, 2018.

[34] Anind K Dey, Katarzyna Wac, Denzil Ferreira, Kevin Tassini, Jin-Hyuk Hong, and Julian Ramos. Getting closer: an empirical investigation of the proximity of user to their smart phones. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 163–172, 2011.

[35] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[36] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[37] Nuno Dionísio, Fernando Alves, Pedro M Ferreira, and Alysson Bessani. Towards end-to-end cyberthreat detection from twitter using multi-task learning. In *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[38] Gregory Epiphaniou, Tim French, and Carsten Maple. The dark web: Cyber-security intelligence gathering opportunities, risks and rewards. *Journal of computing and information technology*, 22(LISS 2013):21–30, 2014.

[39] ESET. Welivesecurity, 2020.

[40] Stack Exchange. The stack exchange data explorer. *Online, http://data.stackexchange.com/. Accessed September*, 2019.

[41] Glenn A Fink, Christopher L North, Alex Endert, and Stuart Rose. Visualizing cyber security: Usable workspaces. In *Proceedings of the 2009 6th international workshop on visualization for cyber security*, pages 45–56. IEEE, 2009.

[42] James B Fraley and James Cannady. The promise of machine learning in cybersecurity. In *SoutheastCon 2017*, pages 1–6. IEEE, 2017.

[43] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009.

[44] Sanjay Goel. National cyber security strategy and the emergence of strong digital borders. *Connections: The Quarterly Journal*, 19(1):73–86, 2020.

[45] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*, 17(3):1294–1312, 2015.

[46] Venkat Gudivada, Amy Apon, and Junhua Ding. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, 10(1):1–20, 2017.

[47] Ayush Hariharan, Ankit Gupta, and Trisha Pal. Camlpad: Cybersecurity autonomous machine learning platform for anomaly detection. In *Proceedings of the Future of Information and Communication Conference*, pages 705–720. Springer, 2020.

[48] JA Hartigan, MA Wong, et al. A k-means clustering algorithm. *New Haven*, 1979.

[49] Aldo Hernandez-Suarez, Gabriel Sanchez-Perez, Karina Toscano-Medina, Victor Martinez-Hernandez, Hector Perez-Meana, Jesus Olivares-Mercado, and Victor Sanchez. Social sentiment sensor in twitter for predicting cyber-attacks using l1 regularization. *Sensors*, 18(5):1380, 2018.

[50] Hiscox. Hiscox cyber readiness report 2020. Technical report, Mcafee, 2020.

[51] Sameera Horawalavithana, Abhishek Bhattacharjee, Renhao Liu, Nazim Choudhury, Lawrence O. Hall, and Adriana Iamnitchi. Mentions of security vulnerabilities on reddit, twitter and github. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 200–207, 2019.

[52] Jack Hughes, Seth Aycock, Andrew Caines, Paula Buttery, and Alice Hutchings. Detecting trending terms in cybersecurity forum discussions. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 107–115, 2020.

[53] Troy Hunt. Have i been pwned. *Last retrieved*, 23, 2019.

[54] Risul Islam, Md Omar Faruk Rokon, Ahmad Darki, and Michalis Faloutsos. Hackerscope: The dynamics of a massive hacker online ecosystem. *arXiv preprint arXiv:2011.07222*, 2020.

[55] Kathleen A Jackson et al. Intrusion detection system (ids) product survey. *Los Alamos National Laboratory*, 1999.

[56] Keenan Jones, Jason RC Nurse, and Shujun Li. Behind the mask: A computational study of anonymous' presence on twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 327–338, 2020.

[57] Aparup Khatua, Apalak Khatua, and Erik Cambria. A tale of two epidemics: Contextual word2vec for classifying twitter streams during outbreaks. *Information Processing & Management*, 56(1):247–257, 2019.

[58] Nitika Khurana, Sudip Mittal, Aritran Piplai, and Anupam Joshi. Preventing poisoning attacks on ai based threat intelligence systems. In *Proceedings of the 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

[59] Eunsoo Kim, Kuyju Kim, Dongsoon Shin, Beomjin Jin, and Hyoungshick Kim. Cytime: Cyber threat intelligence management framework for automatically generating security rules. In *Proceedings of the 13th International Conference on Future Internet Technologies*, pages 1–5, 2018.

[60] Yoohwan Kim, Ju-Yeon Jo, and Kyunghee Kim Suh. Baseline profile stability for network anomaly detection. In *Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)*, pages 720–725. IEEE, 2006.

[61] R Kohavi and F Provost. Confusion matrix. *Machine learning*, 30(2-3):271–274, 1998.

[62] Ba-Dung Le, Guanhua Wang, Mehwish Nasim, and Muhammad Ali Babar. Gathering cyber threat intelligence from twitter using novelty classification. In *Proceedings of the 2019 International Conference on Cyberworlds (CW)*, pages 316–323. IEEE, 2019.

[63] Quentin Le Sceller, ElMouatez Billah Karbab, Mourad Debbabi, and Farkhund Iqbal. Sonar: Automatic detection of cyber security events over the twitter stream. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 23. ACM, 2017.

[64] Kuo-Chan Lee, Chih-Hung Hsieh, Li-Jia Wei, Ching-Hao Mao, Jyun-Han Dai, and Yu-Ting Kuang. Sec-buzzer: cyber security emerging topic mining with open threat intelligence retrieval and timeline event annotation. *Soft Computing*, 21(11):2883–2896, 2017.

[65] Phil Legg and Tim Blackman. Tools and techniques for improving cyber situational awareness of targeted phishing attacks. In *Proceedings of the 2019 International Conference on Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, pages 1–4. IEEE, 2019.

[66] Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. Malwaretextdb: A database for annotated malware articles. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1567, 2017.

[67] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000.

[68] Yizhi Liu, Fang Yu Lin, Zara Ahmad-Post, Mohammadreza Ebrahimi, Ning Zhang, James Lee Hu, Jingyu Xin, Weifeng Li, and Hsinchun Chen. Identifying, collecting, and monitoring personally identifiable information: From the dark web to the surface web. In *Proceedings of the 2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2020.

[69] Steven Loria. textblob documentation. *Release 0.15*, 2, 2018.

[70] Julie Beth Lovins. Development of a stemming algorithm. *Mech. Transl. Comput. Linguistics*, 11(1-2):22–31, 1968.

[71] Pablo Loyola, Kugamoorthy Gajananan, Yuji Watanabe, and Fumiko Satoh. Villani at semeval-2018 task 8: Semantic extraction from cybersecurity reports using representation learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 885–889, 2018.

[72] Check Point Software Technologies LTD. Check point research 2020 cyber security report. *available at https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf. pdf (accessed 25th July, 2020)*, 2020.

[73] Chunping Ma, Huafei Zheng, Pengjun Xie, Chen Li, Linlin Li, and Luo Si. Dm_nlp at semeval-2018 task 8: neural sequence labeling with linguistic features. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 707–711, 2018.

[74] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[75] Kevin Makice. *Twitter API: Up and running: Learn how to build applications with the Twitter API.* O'Reilly Media, Inc., 2009.

[76] Fireeye Mandiant. M-trends 2020 special report. 2020.

[77] R Manikandan, Krishna Madgula, and Snehanshu Saha. Teamdl at semeval-2018 task 8: Cybersecurity text analysis using convolutional neural network and conditional random fields. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 868–873, 2018.

[78] Ericsson Marin, Jana Shakarian, and Paulo Shakarian. Mining key-hackers on darkweb forums. In *Proceedings of the 2018 1st International Conference on Data Intelligence and Security (ICDIS)*, pages 73–80. IEEE, 2018.

[79] Mcafee. Definitive guide to cloud threat protection. Guide, Mcafee, 2020.

[80] Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill. Flow clustering using machine learning techniques. In *Proceedings of the International workshop on passive and active network measurement*, pages 205–214. Springer, 2004.

[81] Sean McKenna, Diane Staheli, and Miriah Meyer. Unlocking user-centered design methods for building cyber security visualizations. In *Proceedings of the 2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2015.

[82] John McQuillan, Ira Richer, and Eric Rosen. The new routing algorithm for the arpanet. *IEEE transactions on communications*, 28(5):711–719, 1980.

[83] Otgonpurev Mendsaikhan, Hirokazu Hasegawa, Yukiko Yamaguchi, and Hajime Shimada. Identification of cybersecurity specific content using the doc2vec language model. In *Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 396–401, 2019.

[84] Robinson Meyer. How many stories do newspapers publish per day. *The Atlantic*, 2016.

[85] Donald Michie, David J Spiegelhalter, CC Taylor, et al. Machine learning. *Neural and Statistical Classification*, 13(1994):1–298, 1994.

[86] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[87] Sean T Miller and Curtis Busby-Earle. Multi-perspective machine learning a classifier ensemble method for intrusion detection. In *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*, pages 7–12, 2017.

[88] Sudip Mittal, Prajit Kumar Das, Varish Mulwad, Anupam Joshi, and Tim Finin. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 860–867. IEEE Press, 2016.

[89] Andrew W Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, 2005.

[90] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *Proceedings of the 2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[91] The Hacker News. Cybersecurity news and analysis, 2019.

[92] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856, 2001.

[93] Eric Nunes, Ahmad Diab, Andrew Gunn, Ericsson Marin, Vineet Mishra, Vivin Paliath, John Robertson, Jana Shakarian, Amanda Thart, and Paulo Shakarian.

Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 7–12. IEEE, 2016.

[94] Philip O'Kane, Sakir Sezer, and Domhnall Carlin. Evolution of ransomware. *IET Networks*, 7(5):321–327, 2018.

[95] Girish Keshav Palshikar, Manoj Apte, and Deepak Pandita. Weakly supervised classification of tweets for disaster management. In *SMERP@ ECIR*, pages 4–13, 2017.

[96] Sergio Pastrana, Daniel R Thomas, Alice Hutchings, and Richard Clayton. Crimebb: Enabling cybercrime research on underground forums at scale. In *Proceedings of the 2018 World Wide Web Conference*, pages 1845–1854, 2018.

[97] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[98] Slobodan Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems*, pages 53–64. Citeseer, 2006.

[99] Peter Phandi, Amila Silva, and Wei Lu. Semeval-2018 task 8: Semantic extraction from cybersecurity reports using natural language processing (securenlp). In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 697–706, 2018.

[100] Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. A rule based approach to word lemmatization. In *Proceedings of IS*, volume 3, pages 83–86, 2004.

[101] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA, 2003.

[102] Marcus J Ranum. A network firewall. In *Proceedings of the World Conference on System Administration and Security, Washington, DC*, 1992.

[103] Adithya Rao and Nemanja Spasojevic. Actionable and political text classification using word embeddings and lstm. *arXiv preprint arXiv:1607.02501*, 2016.

[104] Philip Robinson and Jochen Haller. Revisiting the firewall abolition act. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 10–pp. IEEE, 2003.

[105] Ariel Rodriguez and Koji Okamura. Cybersecurity text data classification and optimization for cti systems. In *Proceedings of the Workshops of the International Conference on Advanced Information Networking and Applications*, pages 410–419. Springer, 2020.

[106] Joshua Roesslein. tweepy documentation. *Online] http://tweepy. readthedocs. io/en/v3*, 5, 2009.

[107] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 135–148, 2004.

[108] Carl Sabottke, Octavian Suciu, and Tudor Dumitraş. Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits. In *Proceedings of the 24th USENIX Conference on Security Symposium*, pages 1041–1056, 2015.

[109] Cyber Safety. Joint select committee—second interim report—cybersafety for seniors: a worthwhile journey. *Adjourned debate on the motion of the chair of the committee (Senator Bilyk)— That the Senate take note of the report*, 2013.

[110] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *Proceedings of the 2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[111] Sagar Samtani, Ryan Chinn, Hsinchun Chen, and Jay F Nunamaker Jr. Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence. *Journal of Management Information Systems*, 34(4):1023–1053, 2017.

[112] Hillary Sanders and Joshua Saxe. Garbage in, garbage out: How purport-edly great ml models can be screwed up by bad data. *Technical report*, 2017.

[113] Anna Sapienza, Alessandro Bessi, Saranya Damodaran, Paulo Shakarian, Kristina Lerman, and Emilio Ferrara. Early warnings of cyber threats in online discussions. In *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 667–674. IEEE, 2017.

[114] Iqbal H Sarker, ASM Kayes, Shahriar Badsha, Hamed Alqahtani, Paul Watters, and Alex Ng. Cybersecurity data science: an overview from machine learning perspective. *Journal of Big Data*, 7(1):1–29, 2020.

[115] Tara Seals. Exodus: New android spyware made in italy, 2019.

[116] Dominic Seyler, Wei Liu, XiaoFeng Wang, and ChengXiang Zhai. Towards dark jargon interpretation in underground forums. *arXiv e-prints*, pages arXiv–2011, 2020.

[117] Han-Sub Shin, Hyuk-Yoon Kwon, and Seung-Jin Ryu. A new text classification model based on contrastive word embedding for detecting cybersecurity intelligence in twitter. *Electronics*, 9(9):1527, 2020.

[118] Ambika Shrestha Chitrakar and Slobodan Petrović. Efficient k-means using triangle inequality on spark for cyber security analytics. In *Proceedings of the ACM International Workshop on Security and Privacy Analytics*, pages 37–45, 2019.

[119] Carson Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny.* CRC Press, 2020.

[120] Rajinder Singh. An overview of android operating system and its security. *Journal of Engineering Research and Applications*, pages 519–521, 2014.

[121] Internet Live Stats. Twitter usage statistics, 2020.

[122] Dave Steinkraus, Ian Buck, and PY Simard. Using gpus for machine learning algorithms. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 1115–1120. IEEE, 2005.

[123] Nan Sun, Jun Zhang, Paul Rimba, Shang Gao, Leo Yu Zhang, and Yang Xiang. Data-driven cybersecurity incident prediction: A survey. *IEEE Communications Surveys & Tutorials*, 21(2):1744–1772, 2018.

[124] I Symantec. Internet security threat report 2019 [j]. 2019.

[125] M Taddeo. Is cybersecurity a public good? *Minds and Machines*, 29(3), 2019.

[126] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the 2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.

[127] Mike Thelwall and Kevan Buckley. Topic-based sentiment analysis for the social web: The role of mood and issue-related words. *Journal of the American Society for Information Science and Technology*, 64(8):1608–1617, 2013.

[128] BK Tripathy, Saurabh Thakur, and Rahul Chowdhury. A classification model to analyze the spread and emerging trends of the zika virus in twitter. In *Computational Intelligence in Data Mining*, pages 643–650. Springer, 2017.

[129] Onur Varol, Emilio Ferrara, Clayton Davis, Filippo Menczer, and Alessandro Flammini. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11, 2017.

[130] ARi Vasudevan, E Harshini, and S Selvakumar. Ssenet-2011: a network intrusion detection system dataset and its comparison with kdd cup 99 dataset. In *Proceedings of the 2011 second asian himalayas international conference on internet (AH-ICI)*, pages 1–5. IEEE, 2011.

[131] Alex Vieane, Gregory Funke, Robert Gutzwiller, Vincent Mancuso, Ben Sawyer, and Christopher Wickens. Addressing human factors gaps in cyber defense. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 60, pages 770–773. SAGE Publications Sage CA: Los Angeles, CA, 2016.

[132] Joseph Vithayathil, Majid Dadgar, and J Kalu Osiri. Does social media use at work lower productivity? *International Journal of Information Technology and Management*, 19(1):47–67, 2020.

[133] DAVID WADDINGTON and MIKE KING. Identifying common causes of uk and french riots occurring since the 1980s. *The Howard Journal of Criminal Justice*, 48(3):245–256, 2009.

[134] Allan H Weis. Commercialization of the internet. *Internet Research*, 1992.

[135] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[136] Ozlem Yavanoglu and Murat Aydos. A review on cyber security datasets for machine learning algorithms. In *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*, pages 2186–2193. IEEE, 2017.

[137] Bassam Zantout, Ramzi Haraty, et al. I2p data communication system. In *Proceedings of ICN*, pages 401–409. Citeseer, 2011.

[138] Feifei Zhang, Jennifer Stromer-Galley, Sikana Tanupabrungsun, Yatish Hegde, Nancy McCracken, and Jeff Hemsley. Understanding discourse acts: Political campaign messages classification on facebook and twitter. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 242–247. Springer, 2017.

[139] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu. Robust network traffic classification. *IEEE/ACM Transactions on Networking*, 23(4):1257–1270, 2015.

[140] Muwei Zheng, Hannah Robbins, Zimo Chai, Prakash Thapa, and Tyler Moore. Cybersecurity research datasets: taxonomy and empirical analysis. In *Proceedings of the 11th USENIX Conference on Cyber Security Experimentation and Test*, pages 2–2, 2018.

[141] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.

[142] Yun Zhou and Peichao Wang. An ensemble learning approach for xss attack detection with domain knowledge and threat intelligence. *Computers & Security*, 82:261–269, 2019.

[143] Yuyang Zhou, Guang Cheng, Shanqing Jiang, and Mian Dai. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Computer Networks*, page 107247, 2020.

[144] Shi Zong, Alan Ritter, Graham Mueller, and Evan Wright. Analyzing the perceived severity of cybersecurity threats reported on social media. *arXiv preprint arXiv:1902.10680*, 2019.