SWINBURNE UNIVERSITY OF TECHNOLOGY

Honours Thesis

Analysing Technique Classification Performance of Cyber Threat Intelligence Extractors

Author:

Hoang Cuong Nguyen

Supervisors:

A/Prof. Bao Quoc Vo

Dr. Shahroz Tariq

Dr. Mohan Baruwal Chhetri

A thesis submitted in fulfilment of the requirements for the degree of Bachelor of Computer Science (Honours)

in the

Faculty of Science, Engineering and Technology

November 2023

TABLE OF CONTENTS

Abstract

In the face of escalating threats posed by emerging cyber-attack techniques, the imperative for rapid and precise information extraction has grown exponentially. Cyber Threat Intelligence (CTI) reports serve as a prominent resource for this purpose, and automatically extracting techniques from CTI reports could further facilitate the decision-making process for cyber analysts and provide resources for further research in new mitigation methods. However, due to the text complexity in the cybersecurity domain and the verbosity of CTI reports, existing CTI extraction programs have suffered from low classification performance and low consistency on replication.

This thesis undertakes a comprehensive assessment of two state-of-the-art CTI extraction programs in terms of technique classification. Inspired by the development of open-source Large Language Models (LLMs), we also evaluated four of these models in terms of classification performance, to test the potential for cross-validation with CTI extractors. Our analysis points out that despite recent progress in technique classification, the performance of CTI extractors remains low. With the high number of false positives, open-source LLMs are not useful for this task either.

In addition, we introduce a novel pipeline for extracting attack techniques from CTI reports. This pipeline involves report summarization using ChatGPT, followed by loading the output into a specially retrained SciBERT model tailored to the cybersecurity domain. Our approach yields promising results: when applied to 10 CTI reports, we observe a substantial 7-percentage-point increase in the F1-score compared to the original SciBERT model for processing entire reports. Furthermore, this research outlines potential avenues for future exploration within the field, aimed at further enhancing the precision of technique extractions.

Acknowledgements

I extend my heartfelt appreciation to those whose support and guidance have been integral to the realization of this thesis.

Foremost, my sincere gratitude goes to Prof. Bao Quoc Vo, my academic supervisor, for his unwavering support and insightful guidance throughout this project. Prof. Vo's expertise in Machine Learning and research skills, coupled with his constructive feedback, has been instrumental in overcoming challenging obstacles and completing this research project.

I am deeply thankful to my industry supervisors, Dr. Shahroz Tariq and Dr. Mohan Baruwal Chhetri from CSIRO’s Data61, for not only accepting me into this project during the Vacation Studentship in summer 2022 but also for their continued guidance during my Honours year. Their mentorship has provided invaluable insights into the realms of cybersecurity, Machine Learning, and open-source projects. Their support has bolstered my confidence in both technical skills and independent research.

Special appreciation is extended to Mr. Yasas Akurudda Liyanage Don, my tutor in the IT Security unit, and his former Master student, Mr. Pajani Mardamootoo. Their expertise in cybersecurity techniques and enhancing Machine Learning models has greatly enriched my knowledge. I am grateful for their guidance and encouragement throughout this thesis.

I express my gratitude to Swinburne University of Technology for facilitating my entry into the Honours course and for continuing my tuition fee scholarship from my Bachelor course.

Acknowledgments are due to CSIRO’s Data61 and its Collaborative Intelligence Future Science Platform (CINTEL FSP) for funding my internship as a Vacation Student in 2022 and providing the opportunity to pursue this research project during my Honours year. I would especially like to thank Dr. Petra Kuhnert for referring me to work as a Vacation Student; and Dr. Jack Collins, previously a Vacation Student himself in CSIRO’s Data61, for providing me guidance on how to choose a research topic for this Honours project.

Lastly, I extend my deepest thanks to my parents for their unwavering love and support. Their financial and spiritual encouragement has been my anchor, motivating me to persevere through challenges and complete this project. Their support has been the driving force behind my quest for an Honours degree, and for that, I am profoundly grateful.

Declaration of Authorship

I, Hoang Cuong Nguyen, declare that this thesis titled, ‘Analysing Technique Classification Performance of Cyber Threat Intelligence Extractors’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Swinburne University of Technology.
- This thesis has been submitted for the degree of Bachelor of Computer Science (Honours) at Swinburne University of Technology, and not to any other institution or for any other qualification.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signature: ___Cuong_______________________

Date: __3/11/2023________________________

List of Abbreviations

CNN	Convolutional Neural Network
CTI	Cyber Threat Intelligence
IOCs	Indicators of Compromise
KG	Knowledge Graph
LLMs	Large Language Models
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
RNN	Recurrent Neural Network
SOC	Security Operations Centre
TTPs	Tactics, Techniques and Procedures

Chapter 1. Introduction

As global connectivity becomes increasingly more important, the risks of cyber-attacks towards governments and institutions worldwide have dramatically increased. To combat these attacks, information about them has been recorded by cyber analysts globally, in the form of Cyber Threat Intelligence (CTI) reports. These reports have been an important source of information for cybersecurity researchers to develop new insights about current threat landscape, and build a proactive defence strategy against future threats [1].

However, analysing CTI reports manually is a tedious task for cyber analysts, since these reports are unstructured and often lengthy (with dozens of pages). Furthermore, security experts still need to rely on their ad-hoc experiences to understand attack behaviours of threat actors [2]. This has caused numerous challenges for Security Operations Centre (SOC) analysts. There are already 70% of analysts who found themselves emotionally overwhelmed due to the amount of attacks they received, and 43% of SOC employees have to turn off attack alerts [3].

Hence, to alleviate the workload burden on SOC analysts, cybersecurity researchers have endeavoured to build a multitude of programs ([4], [5], [6]) designed to autonomously extract essential information from CTI reports, including Indicators of Compromise (IOCs) and tactics, techniques, and procedures (TTPs). Despite commendable strides in this domain, researchers face manifold challenges in establishing a dependable system for extracting threat intelligence, particularly TTPs [7]. These challenges, if left unaddressed, may precipitate an elevated incidence of false positives and false negatives in the outcomes, thereby compromising the overall trustworthiness of the system [8]. There are three key challenges that merit attention:

1. *Domain-level complexity:* CTI reports often incorporate terminology specific to the cybersecurity domain, imbued with meanings and contexts distinct from commonplace English language usage. This discrepancy poses a formidable challenge for Natural Language Processing (NLP) tools trained on conventional linguistic datasets. Furthermore, sentences within CTI reports may lack a clear subject or object, or exhibit omissions in sentence-ending characters (e.g.,), rendering them intricate for widely used NLP toolkits such as NLTK [1], spaCy [2] to analyse.
2. *Verbosity:* CTI reports, by nature, can be extensive, with only a fraction of sentences pertinent to detailing a cyber-attack. For instance, in the case of the DustySky operation report [9] spanning 42 pages, merely 11 sentences convey substantive information about the attack, while the remainder comprises non-incident-related content, such as introductory details.

[1] https://www.nltk.org/index.html
[2] https://spacy.io/

3. *Relationship extraction:* Comprehending the context and information flow between entities within a CTI report is pivotal for obtaining a comprehensive understanding of the tactics and techniques employed in a cyber-attack. However, this poses a formidable challenge for existing NLP systems [10], , necessitating innovative solutions to enhance the efficacy of automated CTI analysis in this regard.

These problems raise a research question: *How could automated CTI extraction programs identify techniques more accurately?* This is the basis for the aim of this Honours project, which is to develop technology to assist cybersecurity personnel to extract relevant knowledge contained in CTI reports, especially the attack techniques classified by the MITRE ATT&CK framework.

In the research and development process for this project, we have made two contributions for the field.

Firstly, we perform a comprehensive evaluation of multiple attack technique extractors, such as AttackG, large language models (LLMs) such as Llama 2 models, and TRAM (Threat Report ATT&CK Mapper). Our evaluation shows that there are significant improvements in performance with in-context learning introduced for CTI extractors, compared with general-purpose LLMs or older classification techniques, such as fuzzy matching. At the same time, we also identified significant pitfalls of innovative models, with the best performing approach only achieving an F1-score of over 0.4, due to overfitting, class imbalances as well as missing techniques in ground truth.

Second and finally, inspired by recent developments in LLMs, we introduce a novel approach to extract techniques from CTI reports. This approach includes two steps: (1) *summarizing* threat reports by GPT-3.5 and (2*) loading reports* into a new SciBERT model for TRAM, retrained with our rebalanced dataset at 4 epochs using in-context learning. During our work for this approach, we found out that one of the main problems contributed to the low F1-score for TRAM is class imbalance, and we resolved this problem by filtering sentences from 50 most prevalent techniques and sub techniques in the MITRE ATT&CK database, out of over 600 of them [11] in a new dataset. This dataset is based on another dataset that was annotated by cybersecurity researchers (CTI-to-MITRE with NLP [12]), but has been filtered and rebalanced, using GPT-3.5, to improve the performance for technique classification.

Our approach has managed to solve the class imbalance problem for current technique extraction programs, as well as achieve higher F1-score than using the full CTI report, with original configuration of TRAM by seven percentage points. We also provide future research directions in the CTI extraction, especially attack techniques extraction, for other researchers.

The remaining part of this thesis will be presented as below. Chapter 2 will be about the current progress in CTI analysis and extraction and current limitations. Chapter 3 is about how our evaluation was set up, and our proposed approach will also be discussed there. Chapter 4 describes the results from our experiments, while chapter 5 is about further discussion of the results. Finally, chapter 6 will mention our research's limitations as well as provide future directions for cybersecurity researchers in this topic.

Chapter 2. Research background

2.1.DEFINITION OF CYBER THREAT INTELLIGENCE

Cyber Threat Intelligence is, primarily, a preventative defence mechanism against cyber-attacks. According to National Institute of Standards and Technology (NIST), it is defined as “threat information that has been aggregated, transformed, analysed, interpreted, or enriched to provide the necessary context for decision-making processes” [13].

One of the primary criteria for evaluating CTI-related information is its actionability, which can be defined as whether cyber analysts can use CTI to combat cyber threats. According to Pawlinski et al. [14], , actionable CTI must have the following characteristics:

- **Relevance:** CTI must be applicable to the area of responsibility for recipients.
- **Timeliness:** CTI information must not be older than a predetermined threshold (in some cases, if CTI is older than a few hours then it needs to be discarded), though too much CTI arrive at the same time could overwhelm the process ability of SOC analysts (as mentioned in section I).
- **Accuracy:** Recipients should get CTI information that is verified and has no error (this could depend on local context).
- **Completeness:** Actionable CTI should provide information for recipients to understand about past cyber-attacks.
- **Ingestibility**: CTI should be shared in a format that could be processed by recipients’ information systems.

Other characteristics for actionable CTI also include priority, implementation, the trustworthiness of the source, relevance to the industry, clear guidance to resolve the threat, and sufficient context [15].

Numerous sources serve as potential inputs for the Cyber Threat Intelligence (CTI) extraction process. Rahman et al. (2023) [1] have meticulously categorized seven distinct types of inputs, namely: social media, darknet forums, threat reports, online articles, vulnerability databases, software manuals, and security literature. In the context of this thesis, our analytical focus will be directed towards programs that specialize in the analysis of data derived from threat reports. Additionally, we will extend our assessment to programs that exhibit proficiency in summarizing other types of inputs, especially those characterized by high textual structural similarities, such as social media content or web text. The analysis will encompass instances where these programs operate in tandem with threat reports, fostering a comprehensive understanding of their integrative capabilities.

2.2. CURRENT PROGRESS IN CYBER THREAT INTELLIGENCE EXTRACTION AND SHARING

Currently, CTI extraction programs include the following steps:

- Identifying extraction sources (threat reports in this case).
- Report crawling.
- Text processing and data labelling.
- Text summarisation (with learning approaches).
- Processing outputs.

The schema for this pipeline is available in Figure 2.2.1. In subsections from 2.2.1 to 2.2.5, each step in this pipeline would be analysed by how current CTI extraction programs do it.

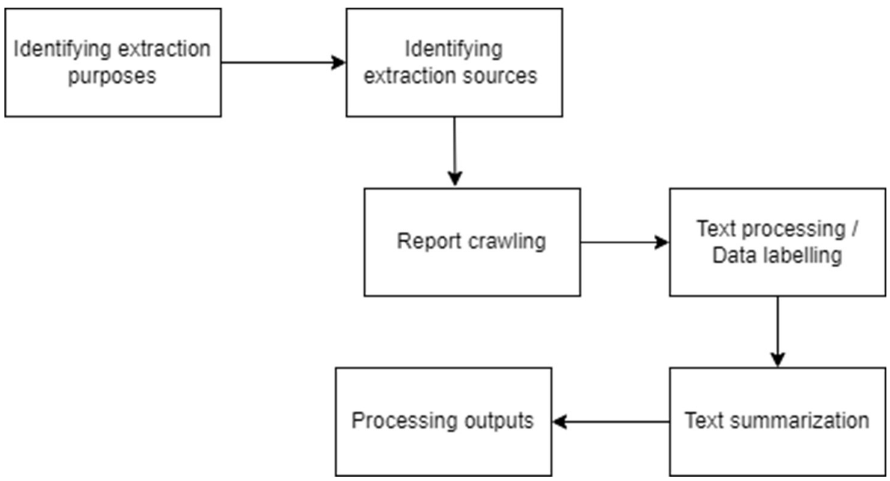

Figure 2.2.1. Steps from a typical CTI extraction pipeline.

2.2.1. CTI extraction purposes and sources

For current purposes to extract CTI data, the main goal for producing knowledge graphs currently is to extract cyber threat relevant entities and relationships from the CTI reports. Entities considered relevant by cyber analysts could include the names of malware, vulnerabilities, software, malicious actors, etc. depending on the types of inputs and the purposes of the programs. For example, Joshi et al (2013) [16] extracted vulnerability information from Common Vulnerabilities and Exposures dataset (CVE), as well as threat-related data (like IOCs) from security blogs, similar to CyGraph [17] or RelExt [18]. Only the most recent programs (AttackG [4], LADDER [5], aCTIon [19]) provide additional information, in the form of tactics, techniques and procedures. It is worth noting that these programs use the MITRE ATT&CK framework to identify techniques as well as tactics.
For threat-related data extraction, analysts could use several sources:

- Security blogs from individuals, communities or institutions in cybersecurity domain, for example Joshi et al [16], or CyGraph [17].
- Security bulletins created by companies such as Adobe, Microsoft; the examples are programs of Piplai et al [20], or RelExt [18].

- Social media posts, by cybersecurity vendors and reputable individuals and organizations in the cybersecurity field, such as the case of CyberTwitter [21], and Cyber-All-Intel [22].
- Threat reports made by large cybersecurity vendors (for advanced persistent threats, vulnerabilities and exploits), such as the case of Extractor [7], ThreatKG [23], and LADDER [5].

It is worth noting that all data sources extracted for CTI summarization pipelines are in form of text. Apart from CyTAG (which also analyses threat-related data from images), no programs have been developed yet to analyse other forms of data.

2.2.2. Report crawling

After identifying extraction sources, researchers usually need to crawl their needed reports from the Internet before training the models. While this task could be done manually (after identifying the dataset needed, such as National Vulnerability Database (NVD), CVE or MITRE's[3] dataset of threat reports), this approach is error-prone and time-consuming. As such, researchers have automatically crawled CTI reports by using web-based crawlers. For example, in the case of CyberTwitter [21], a crawler based on searching keywords from Twitter Stream API is deployed to identify CTI-candidate text. In AttackG, ThreatKG [23], CSKG4APT [24], LADDER [5], researchers deployed an HTML-based web crawler to extract CTI reports.

2.2.3. Text processing and data labelling

- **Step 1: Text processing:** For this step, the textual data collected needs to discard punctuation marks, URLs, irrelevant symbols and stop words, as well as correct the spellings. Two other steps that must be applied are tokenization and lemmatization, to break sentences into words one by one. These steps aim to homogenize the textual data and prepare for using NLP techniques. Popular libraries for these tasks are NLTK from Python, spaCy (in Extractor and AttackG), as well as large language models (LLMs) such as BERT, RoBERTa (in the case of LADDER), or GPT-3.5 (with aCTIon [19]). In the case of CyTAG, the program could collect and process images due to the application of the *tesseract* library in Python [25].
- **Step 2: Data labelling:** An important task for researchers to build and evaluate CTI extraction pipeline is labelling data into training sets and test sets. Correct labelling of CTI-related texts is important, since it would determine the accuracy of summarizations, and therefore determine whether CTI could be considered as actionable or not. However, since there is no commonly labelled dataset as a benchmark, researchers must either label their reports themselves, or deploy annotators with experience in cybersecurity, such as graduates or cybersecurity experts. Human-annotated data are then used as a benchmark to evaluate the accuracy of summarization, such as in the case of AttackG, where the data annotated by experts are considered as having neither false positive nor false negative.

[3] https://attack.mitre.org/software/

2.2.4. Text summarization

- **Part-of-Speech Tagging:**

For this method, each word in the sentence will be mapped with a Part-of-Speech (POS), such as adjective, noun or verb. This technique is useful for finding passive sentences to convert into active ones, to homogenize the report (such as in Extractor program [7]). Since from the semantic role of a word in a sentence, we could identify the dependencies and grammatical structure in the sentence; this method could be used simultaneously with dependency parsing to improve the accuracy of this method.

- **Dependency Parsing:**

For this method, the dependencies between words in a sentence would have to be analysed, to examine the grammatical structure of a sentence (whether a word belongs to subject, verb, or object). This method is currently being used to identify attack patterns from CTI-candidate text, by extracting malicious actors (subjects), attack behaviours (verbs), and victims (objects), as given in the Extractor [7], CyTAG [25], Cyber-All-Intel [22] and Piplai et al. (2020) [20] . It is also worth noting that the Stanford library for dependency parsing[4], as one of the most popular frameworks for this task, is used in all three programs (even in the case of [20], for which researchers developed their own Malware Entity Extractor (trained with their own dataset for malware, rather than using the original library), they still used the same library from Stanford). This is due to the ability of the parser to represent both the syntax and the semantic structure in English ([26]).

- **Coreference Resolution:**

Any parts of the report that mention the same entity will be merged. This technique is used to reduce the output' size and to connect disjointed parts of the graph together into a whole attack graph. In our selected programs, Extractor [7], ThreatKG [23] and AttackG [4] are the most prominent examples of the implementation of this technique; with libraries such as *neuralcoref* and *coreferee* being used. However, this technique is only necessary if the output is a visualised graph; for the case of programs producing ontologies, such as Mulwad et al. [27], or only generated triplets as an intermediary step like LADDER[5], this method is not implemented.

- **Named Entity Recognition (NER):**

This method is about classifying words into distinct types of entities, and this can be applied to identifying types of malicious actors, such as vulnerabilities, services, files, executables, etc. Various CTI extraction programs implement this method in their task, including threat reports extractors such as Extractor [7], AttackG, ThreatKG [23], Open-CyKG [28] and non-threat reports analyzers, such as from Mulwad et al. (2011) [27], RelExt [18] and CyberTwitter [21]. A point worth noting is that researchers could either use existing models for this task, such as spaCy (for Extractor, AttackG), OpenCalais (in the case of the program from

[4] https://nlp.stanford.edu/software/stanford-dependencies.html#English

Mulwad et al.), or building their own extractor that is trained with security text, such as a model based on BERT with the case of CSKG4APT [24].

- **Machine Learning Models:**

Although there are multiple learning approaches (supervised, unsupervised and reinforcement learning), in the context of our selected programs, the only ML approach used is supervised learning. For this method, models would be trained with labelled data before applying to a corpus of unlabelled texts. This task could use multiple features, such as Term Frequency–Inverse Document Frequency (TF-IDF), and word embedding. Finally, these features are parsed to supervised models to classify features to the corresponding models.

Supervised classification is used to:

- Recognize the relationship between two different entities, and update the knowledge graph (such as the case of RelExt [18]);
- Recognize whether a particular segment of texts matches any definition of attack techniques (sentence by sentence, in the case of LADDER, or by constructing technique templates and assign a matching score threshold, in the case of AttackG). It should be noted that this approach is also used by other TTPs extraction programs, such as TTPDrill [29].
- Recognize whether the text is relevant in terms of CTI, or contains any data related to cybersecurity events, such as the studies of Mulwad et al [27], and Joshi et al [16].

For this classification task, the most prominent technique used is neural networks (CNN, RNN, graph neural networks). Types of neural networks used include transformers, language models (like BERT, RoBERTa), long-short term memory (LSTM, BiLSTM), conditional random fields (CRFs) and Gated Recurrent Units (GRU, in the case of Open-CyKG and CyberRel). Support vector machines (SVM) are also being used, in the case of Mulwad et al [27]. While pre-trained models could be used directly, such as the case of BERT in CyberRel, the current trend for CTI extraction programs is to further train these models with cybersecurity-related corpus.

- **Other techniques:**

Although NLP techniques as mentioned above are increasingly being used in CTI summarization pipeline, there are programs that do not use any of these techniques, as well as ML models. A prominent example is CyGraph [17], which ingests the data through REST API, before using ANTLR (Another Tool for Language Recognition) [30] to translate the data to its query languages (CyQL), and process the data in a graph database (Neo4j). Joshi et al [16] used IDS ontologies to represent vulnerabilities and other concepts (linked with DBpedia), and the same method is also used by Mulwad et al [27]. Apart from using their own hierarchical ontology to represent threat knowledge, ThreatKG [23] is prominent in their use of rules for regular expressions to extract and count the number of IOCs, and using data programming to automatically annotated the dataset for training. Mulwad et al. also use BRAT[5] library to rapidly annotate their dataset. For populating the graph, Cyber-All-Intel [22] used 'Relational Concurrence Assumption' principle to train the vectors before embedding them and fill the knowledge graph with these vectors.

It should be noted that since most of the CTI information is currently written in English, current CTI summarization pipelines are trained and could only run with reports written in English. The exception is CSKG4APT [24], which was trained with reports in Chinese, and achieved a F1-score of 81.88% with inputs in this language.

2.2.5. Processing outputs

A trendy way to generate outputs for CTI extraction programs is to create knowledge graphs. They could be represented as $G = (E, R, S)$ (according to Liu et al. [31]), in which:

- E is a set of entities: $E = \{e_1, e_{2,} e_3, \ldots, e_N\}$ (categories and its objects (system, service, vulnerability, malware, etc.) as well as IOCs and object types)
- R is a set of relationships: $R = \{r_1, r_{2,} r_3, \ldots, r_N\}$ (links between entities)
- Each entity in E could have connections with each other using triplets (S) with the format <entity, relationship, entity>, for example $< e_1, r_{1,} e_2 >$.
- There are also attributes, or characteristics for entities such as tactics, techniques, and procedures (TTPs) corresponding to entities; links to outside sources such as to Google.com, DBpedia[6], etc.

There are multiple ways to export a knowledge graph:

- *As an ontology:* Since ontologies are defined similarly as knowledge graphs (as a set of classes, attributes and relationships used to model a domain of knowledge [32]), they are an ideal method to represent graphs for output. There are two types of ontologies developed for CTI sharing:
  + **Resource Description Framework (RDF)**[7]: a standard model for exchanging data on the Web. Programs producing output in RDF format include Joshi et al [16] and Mulwad et al [27].
  + **Unified Cybersecurity Ontology (UCO)** [33]**:** an ontology based on STIX to facilitate data sharing. It is worth noting that there are multiple versions of UCO since it was released in 2017, the latest one being 1.2.0 [8]. This ontology is not generated as the final output, but rather as an intermediate step to identify types of entities in RelExt [18]
  + **OWL Web Ontology Language [9]:** This is an ontology for Semantic Web with meaning being formally defined. Current examples with OWL include Piplai et al [20] and Mulwad et al [27] (for the purposes of reasoning).
  A similar case with ontology is knowledge triplets, which is generated in the case of LADDER before being used to infer attack techniques.

[5] https://brat.nlplab.org/
[6] https://www.dbpedia.org/
[7] RDF - Semantic Web Standards (w3.org)
[8] https://unifiedcyberontology.org/releases/archive/
[9] https://www.w3.org/TR/owl2-primer

The main advantage of this approach is that CTI information generated could be shared to multiple parties if receivers also understand the same ontology (IDS or RDF). However, their ontologies are not user-friendly, and therefore the output would need to be processed by visualisers like VisGraph3[10] or Ontospy[11].

- *As a displayed graph:* Most of our selected research generate graphs as an output. One similarity between them is that in all cases, graphs produced are directed graphs, meaning that we can know the sequence of actions happening in a cyber attack (except for the case of AttackG, which does not have a directed graph or numbered edges in the first place).
  One of the important differences for graph generation is how programs represent entities in cyber-attacks: Extractor [7], for example, has four types of entities: file, registry, process and socket; while AttackG [4] has nine types of entities. Both programs represent them as different shapes, while ThreatKG uses distinct colours to distinguish them in their website.
  A point worth noting is that while the graph generation process for all programs follows similar steps (implementation, type selection and improving), as recommended by [34], currently after finishing these steps, no programs have offered users a way to request changes in output generated by summarization models.

However, outputs generated by CTI extraction programs could also be represented by other ways, such as using popular standards and protocols for CTI sharing, like Structured Threat Information Expression (STIX)[12], OpenTPX (Threat Partner eXchange)[13], Malware Attribute Enumeration and Characterization (MAEC)[14]. Examples of programs sharing threat intelligence in this way include aCTIon [19], and AttackG [4] (for sharing templates found in MITRE ATT&CK database).

2.3. CURRENT LIMITATIONS OF CYBER THREAT INTELLIGENCE EXTRACTION PROGRAMS

While currently various methods have been successfully developed to detect CTI attacks, there are many challenges that these models are currently facing:

- **Lack of a clean, label and published dataset:** Most of the datasets used for training these programs are crawled and labelled manually by authors, or with help of cyber security professionals. There are currently no shared datasets for CTI research; even though MITRE provided their own resources for techniques in cyber-attacks, each attack has multiple reports rather than one, so it is hard to find original reports as the benchmark.
- **Dependency explosion:** This is the phenomenon in which every output of a process could become input of another operation. For long CTI reports, the

[10] https://visgraph3.github.io/
[11] https://pypi.org/project/ontospy
[12] https://stixproject.github.io/about/
[13] https://github.com/Lookingglass/opentpx/
[14] https://maecproject.github.io/

running process of a summarization program could be prolonged, resulted in a huge, illegible graph. This could also lead to false positives, since the program might attribute an entity that did not participate in an attack as a malicious actor [35].

- **Lack of a cybersecurity custom NER model:** While recent models have either incorporate cybersecurity vocabularies in their training corpus (Extractor and CyTAG as extension), many programs still used generic models for NER and dependency parsing (NLTK, Stanford NER) without modifying their models for cybersecurity-related words (for example, sec2vec).
- **Lack of a mechanism for human-AI collaboration:** Currently, users could not modify the knowledge graph generated by the summarization model, and there is only one program, ThreatKG [23] that allow users to interact with the provenance graph. This could hinder the efficiency of CTI summarization models, since analysts could not apply their knowledge to reduce false positives and false negatives from the result, and therefore might fail to identify the real and most urgent threats [36].
- **Low consistency on replication:** A problem for numerous CTI extraction pipeline is that their performance (as claimed in the paper) could not be replicated when tested with another dataset. For example, although AttackG's authors claimed that their program's F1-score is 78.9%, when testing against LADDER its score dropped to 15% compared to 64% from LADDER [5]. The reason could be due to the difference in dataset and testing conditions (for this task LADDER used five malware reports rather than sixteen reports like AttackG). This could potentially lead to duplication of efforts and low actionability for CTI analysis.

Chapter 3. Evaluating approaches for extracting attack techniques from CTI reports

3.1. OVERVIEW

In this part, we would explain how we set up the analysis of attack technique extraction platforms, to answer the research question about how to extract techniques from threat reports more accurately.

Overall, for this evaluation, we have conducted a quantitative analysis of open-source CTI extraction programs that could generate attack techniques from the MITRE ATT&CK framework. We have evaluated the performance of these programs, based on datasets of CTI reports that have techniques being manually annotated by cybersecurity experts. These CTI reports have been considered as ground truth, and we have assessed these programs based on:

- The number of techniques that were correctly marked by these programs.
- The number of techniques that were incorrect or were not identified by these programs.

This information would be the basis for the classification metrics that we used for the evaluation in this thesis.

We would discuss about four distinct factors in our evaluation:

- What CTI extraction programs we chose for evaluation.
- What was the ground truth for our analysis of technique extraction performance.
- What tests did we conduct for CTI extraction programs.
- What metrics did we use to evaluate these programs.

These factors would be analysed in Sections 3.2, 3.3, 3.4 and 3.5, respectively.

3.2.CTI EXTRACTION PROGRAMS FOR EVALUATION

To evaluate existing programs, we need to establish criteria for choosing current programs for extracting attack patterns:

- The output from these programs need to contain techniques from the MITRE ATT&CK Enterprise framework; since this framework has become the most widely adopted framework for modelling cyber-attack scenarios and strengthen security posture for institutions [37].
- These programs need to be open-source and could be deploy on a local server.
- These programs should be developed and deployed recently (from 2022 onwards).

In the end, there are two programs that we chose to evaluate:

- AttackG [4], a program that generates knowledge graphs with technique annotations from the whole MITRE ATT&CK database;
- TRAM (Threat Report ATT&CK Mapper) [6], a program listing potential techniques found in CTI reports. This program is trained with 50 most popular attack techniques and sub techniques from MITRE ATT&CK database [11].

Both programs were recently developed (AttackG in 2022, newest version of TRAM in 2023), therefore they could be regarded as state-of-the-art programmes for extracting attack techniques. Additionally, we tested two other programs:

- LADDER [5], which is a technique extractor from CTI text and was developed and released in 2023.
- Extractor [7], which is also a program that analysing CTI report and return knowledge graph (like AttackG), though it does not return attack techniques.

However, we did not conduct a comprehensive evaluation of these two programs since:

- For LADDER's case: this program only returns a list of techniques, as displayed in figure 3.1.1 (rather than a knowledge graph like in the case of AttackG) and was trained with the whole MITRE ATT&CK database (rather than a part of it in the case of TRAM [11]), and thus could not fit to any of our tests.

```
T1534 (0.5205462127923965, 'Carbanak has taken advantage of system users by launching spearphishing
attacks in order to get their malware on target.') Internal Spearphishing
T1588 (0.532364621758461, 'Carbanak has abused the trust of digital signatures by creating a fake
identity in order to obtain valid certificates from a certification authority (CA)4 for their variant of
the Anunak malware, which is also called Carbanak.') Obtain Capabilities
T1059 (0.564177542924881, 'In addition to custom malware, Carbanak has been known to use administrative
tools native to the Windows environment, including PowerShell, WMI, and RDP.') Command and Scripting
Interpreter
T1189 (0.5419586896896362, 'Carbanak is reported to begin most breaches with spearphishing and social
engineering in order to get a legitimate user to download a Microsoft Word document with malicious files
embedded in the document.') Drive-by Compromise
T1608 (0.5159487575292587, 'They are also known to host malicious files on Google Docs and PasteBin to
further expand their command and control.') Stage Capabilities
T1586 (0.49516957998275757, 'Once on target, Carbanak has been found to rely on using valid accounts to
perform most of their actions.') Compromise Accounts
T1591 (0.5679598897695541, 'The group is known to move laterally and escalate their privileges across
networks to find critical systems that manage financial transactions.') Gather Victim Org Information
T1014 (0.5336565524339676, 'Carbanak has been found to target hosts that have specific banking software
that would facilitate the illicit funds transfers.6 The group is reported to then establish persistence
using Windows native tools, such as scheduled tasks and auto-run services, or other non-malicious tools,
such as VNC.') Rootkit
Duration: 0:01:05.253999
```

Figure 3.1.1. Results generated by LADDER for the Carbanak intelligence summary [38].

- For Extractor's case: since we did not get a trained Machine Learning model, as well as a dataset for retrain the model that the program is based on, we could not reproduce the result like the Extractor paper. We will discuss more about problems that we faced in part 5.1.

We would discuss more about the results from LADDER and Extractor in chapter 5 (Discussion) of this thesis.

Given the rising popularity of Large Language Models (LLMs) recently, we also want to evaluate LLMs in terms of attack techniques extraction. This could serve as a means for cross-validation, as well as providing a basis for humans to modify the outputs generated by CTI extraction programs above. For that, we have several criteria that LLMs need to follow:

- LLMs need to be open-source (so that the program would be free to use, as well as to fine-tune with CTI-related datasets).
- Ideally, LLMs should be able to run in basic versions of Google Colab notebooks, with a T4 GPU (so that users could run the model for free).

As a result, we choose h2oGPT[15] for our analysis, since this website provides a way to run multiple LLMs simultaneously, with a low amount of time. LLMs that we choose to analyse from h2oGPT are h2oai/h2ogpt-gm-oasst1-en-2048-falcon-7b-v2, Llama-2-70b-chat, Llama-2-13b-chat, and Llama-2-7b-chat. All these models are fine-tuned for question answering, though we only used the original versions of these

[15] https://gpt.h2o.ai/

chatbots (without fine-tuning for CTI domain). Although only Llama-2-13b-chat and Llama-2-7b-chat could be run for free in basic versions of Google Colab [39], for other models we could use the website of h2oGPT to run it for free. [16]

3.3.DATASETS AS GROUND TRUTH

Given that one of the biggest limitations for CTI analysis is that there is not a clean and well-labelled dataset in terms of TTPs for CTI domain, one of the first tasks needed for this project is to choose a well-annotated dataset. The methodologies for building such a dataset are:

- Datasets need to include CTI reports, with clearly labelled techniques from the MITRE ATT&CK framework. Ideally, the ID of each technique should also be included, to simplify our reference.
- The reports should be in the whole (so there would be less risk of false negatives, due to difference in annotations between different cyber analysts).
- Each report should also have four thousand words at most, to fit with the current limits of Large Language Models (LLMs).

Ultimately, our selection encompassed six reports sourced from the Adversary Emulation Library, a repository documenting emulation endeavours for various cyber-attack campaigns undertaken by MITRE Engenuity's team [40]. The chosen reports pertain to well-known attack campaigns, specifically APT29, Carbanak, FIN6, FIN7, menuPass, and OilRig. The rationale behind opting for these reports is multi-faceted:

- *Conciseness:* The selected reports exhibit brevity, ranging from 165 words in the case of the OilRig report to 423 words for the FIN6 report. This characteristic facilitates the summarisation process by contemporary CTI extraction programs.
- *Annotation Clarity*: Notably, these reports boast clear and comprehensive annotations of attack techniques, with each technique accompanied by its corresponding ID from the MITRE ATT&CK framework. This meticulous annotation aids in establishing a robust ground truth for the evaluation process. An example is displayed in figure 3.2.1, illustrating the explicit linkage between attack techniques and their respective IDs within the CTI reports.

[16] When this thesis is written (October 2023), the model of h2oai/h2ogpt-gm-oasst1-en-2048-falcon-7b-v2 is not available anymore in h2oGPT (The test was conducted in August 2023). However, the model is still available in HuggingFace: https://huggingface.co/h2oai/h2ogpt-gm-oasst1-en-2048-falcon-7b-v2

Operations: OilRig's tradecraft is notable for their commitment to stealth and persistence in the pursuit of their mission objective. OilRig has demonstrated adeptness in a broad range of attack vectors and a willingness to deviate from their existing arsenal to use novel techniques to accomplish their objectives. OilRig predominantly leverages social engineering as an initial attack vector but has also exploited patched vulnerabilities. Following the 2019 leak of OilRig's tools and victim data, the group actively evolved their payload arsenal and retooling to avoid detection, creating multiple different malware variants with the same purpose as always: to gain an initial foothold on targeted devices. [7 8]

OilRig is reported to have attained initial access via phishing (T1566.001,T1566.002, T1566.003), and credential abuse (T1078). [8 9] After achieving an initial foothold, OilRig actors are reported to use credential dumping tools, such as Mimikatz, to gather legitimate account credentials (T1003.001) then move laterally to other systems on the network. OilRig is also known for installing webshells to transfer tools (T1105) and maintain persistent access to the network. [10]

Figure 3.2.1. OilRig's intelligence summary (from [41]).

Another dataset that we used for this analysis is Attack-Technique-Dataset from NewBee119 [42]. Four attack reports from this dataset are chosen, from past campaigns such as OceanLotus [43], Sowbug [44], MuddyWater Operations [45], and LuckyMouse [46]. These reports are lengthier than the reports from Adversary Emulation Library (with MuddyWater Operations having 1842 words, for example), and thus testing CTI programs with these reports could give us useful insights about how these programs could summarize and find TTPs from extensive attack reports typically encountered in cybersecurity vendor documentation, which often spans numerous pages.

A pertinent consideration for our second test involves the refinement of the ground truth. Given that TRAM is trained using the fifty most common techniques in the MITRE ATT&CK framework, certain adjustments were undertaken:

- *Adversary Emulation Library:* We omitted tactics that TRAM had not been previously exposed to during its training.

- *Attack-Technique-Dataset Reports:* An iterative process was employed to align the current approaches with the IDs in the dataset, acknowledging that the dataset's publication occurred four years ago, and the MITRE ATT&CK database has undergone subsequent updates. Techniques which were not present in the list used for TRAM's training were consequently excluded.

Technique labels in the Adversary Emulation Library serve as the foundation for evaluating AttackG and other LLMs. Appendix A contains the ground truth following processing for the TRAM assessment.

For each report in the combined dataset, we performed various steps of preprocessing:

- For intelligence summaries (Adversary Emulation Library): We chose the operations (the part described the cyber incident step by step) as the input for CTI extractors. Then we remove technique IDs, as well as hyperlinks to reports, so that CTI extractor programs would not know about technique IDs before.
- For reports from Attack-Technique-Dataset: Since these reports are longer, we would remove the headlines and the lists of IOCs, which are commonly shown at the end of the report, because we could not infer techniques from these parts of the reports and could confuse CTI analysis pipelines.

Since the dataset for the analysis of TRAM is larger compared to the analysis of AttackG and other LLMs, the main limitation of our dataset choice is that we could not compare the performance between all these three programs directly (which is due to difference between sets of attack techniques that they were trained on). Nevertheless, we could still understand about the extraction performance of these programs in different threat reports.

3.4.IMPLEMENTED TESTS

Given CTI extraction programs that we chose for our evaluation are trained with different sets of techniques (AttackG and current open-source LLMs are trained with the whole MITRE ATT&CK database, while TRAM was trained with only fifty most prevalent techniques in MITRE ATT&CK framework), we decided to conduct three different tests:

1. Comparison between AttackG and open source LLMs.
2. Comparison between different approaches for TRAM's deployment.
3. Comparison between potential techniques to improve TRAM's performance.

All approaches' implementations are discussed in parts 3.3.1, 3.3.2, and 3.3.3, respectively. We would also discuss our proposed strategy in section 3.3.4 because it is based on improving TRAM's existing machine learning models.

3.4.1. Comparison between AttackG and open-source Large Language Models.

As we discussed in part 3.2, AttackG is a program that summarizes a CTI report and generates a knowledge graph that includes attack-related entities (IOCs, vulnerabilities, software, systems, and so on) and their dependencies, as well as potential attack techniques. AttackG's output is presented in Graphviz [17] format, which is then illustrated in a PDF file. Each attack technique is then represented as a subgraph in the technique knowledge graph (as shown in figure 3.4.1).

[17] https://graphviz.org/

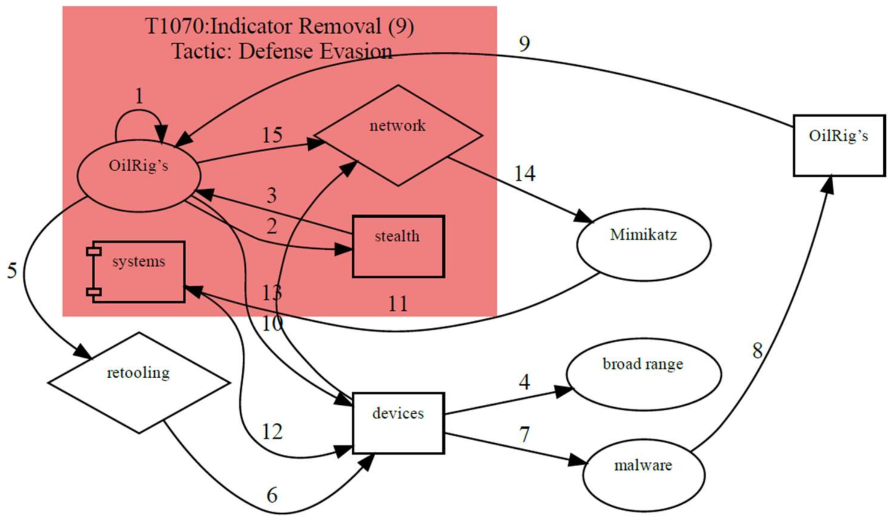

Figure 3.4.1. The original technique knowledge graph generated by AttackG, for OilRig report [41]. Note that only one technique, T1070, could be displayed even though AttackG identified 24 techniques for this report, since many techniques are overlapped with each other and Graphviz does not allow overlapped techniques (as subgraphs) to be displayed as the same time. [47] It should be emphasized that in this graph, the ovals, rectangles, diamonds, and books represent executables, files, networks, and systems, respectively.

Since there could be multiple attack techniques with the same entities found by AttackG, our approach to tackle this problem is:

- Building a user interface for AttackG. This would allow users to copy and paste the report and send it to the AttackG model in the backend (which is hosted with Flask), using REST API. The graphs are then visualised on the website, using d3-Graphviz library [18].
- Implementing a function in the backend to separate techniques with similar entities and placed them into new files on Graphviz.
- Creating more functionalities to set up the default knowledge graph for users, as well as save the user' choice for attack techniques in the local server.

To assess our approach, we would investigate the original graphs generated by AttackG, to find whether the program could find accurate techniques from a CTI report as much as possible; since if AttackG has high true positives and low false positives, there would be less versions of graphs generated by the program, but with much higher confidence by users.

As for Large Language Models, we used zero-shot prompting to identify attack techniques, by asking the chatbots to write the list of techniques, as well as their corresponding IDs, from the MITRE ATT&CK framework. If both AttackG and one of the LLMs have high classification performance (especially in terms of precision and

[18] https://github.com/magjac/d3-graphviz

F1-score, since we want to limit the number of false positives, for reasons as we mentioned in part 2.3), we would incorporate the new LLM into our website of AttackG for cross-validation, and as a basis for add/remove techniques from AttackG's result.

3.4.2. Comparison between different approaches for TRAM's deployment.

In the newest version of TRAM, the program used the SciBERT model [48], which is a BERT (Bidirectional Encoder Representations from Transformers) fine-tuned for scientific texts. This model was trained (using in-context learning) with a labelled dataset from CTI reports [19] to classify sentences to techniques in the list of fifty most prevalent ones. Two parameters could be modified to change the result for each CTI report:

- The threshold of sentences that each technique contains.
- The threshold of confidence for SciBERT in a technique, for each sentence (the original threshold value is 25%). For the purposes of the evaluation, we will evaluate difference in TRAM's performance between 25% and 80% level of confidence.

Inspired by the implementation of aCTIon, we also tried to summarize threat reports, by asking GPT-3.5 to find sentences containing techniques from these reports. Our prompt used for this task is:

```
Use the following portion of a long document to see if any
of the text is relevant to answer the question. Return any
relevant text verbatim.
(Threat report)
Question: Which techniques are used by the attacker?
Report only relevant text, if any.
```

We would record the summary and assess the performance of the originally trained SciBERT model, with confidence levels of 25% and 75%, respectively. This is based on aCTIon's evaluation steps, and based on our experience, we believe that increasing confidence to 80% for summarised reports would result in an excessively low number of identified techniques—especially for short reports—and therefore lower reliability (since users typically expect an attack campaign to be carried out using multiple techniques).

Thus, for this comparison, we would conduct tests based on four different configurations:

- The original configuration of TRAM (originally trained SciBERT model, at 25% confidence level).
- The original trained SciBERT model, with higher confidence threshold (80%).
- Using the summarization approach of aCTIon for the original TRAM website (25% and 75% confidence levels, respectively).

[19] https://github.com/center-for-threat-informed-defense/tram/blob/main/data/tram2-data/single_label.json

3.4.3. Comparison between techniques to improve TRAM's performance.

Various techniques are commonly used to improve the classification performance of an LLM, such as fine-tuning the model with another dataset, rebalancing the amount of data between classes for in-context learning, and deploying k-fold validation. These methods are crucial for raising the performance of TRAM, especially since the structure of the dataset provided by TRAM's authors is heavily imbalanced (in figure 3.3.2). For these approaches, we would use the original pipelines for 2023 version of TRAM, provided in form of Google Colab notebooks.

The SciBERT model used for this task has the same configuration with the original BERT uncased model (which has twelve attention layers, twelve attention heads, 768 hidden dimensions, and maximum context length as 512).

During this study, comprehensive training and fine-tuning procedures were executed on the complete SciBERT uncased model, enabling the adjustment of all model parameters to suit the characteristics of the new dataset. Although the SciBERT model [20]is pre-trained for scientific text and encompasses functionalities tailored for text classification, its proficiency in our task is acknowledged despite its lack of domain-specific knowledge in cybersecurity, a feature present in alternative models such as SecBERT [49] and SecureBERT [50]. Noteworthy is the fact that the latter models exclusively possess capabilities related to word completion and mask filling.

Throughout our experimentation, encompassing in-context learning scenarios with and without k-fold validation, as well as fine-tuning iterations, the AdamW optimizer was consistently employed. The specified optimization utilized a learning rate of 2e-5 and ε value of 1e-8, omitting bias correction. The contextual comprehension capacity of the SciBERT model was constrained to 512 tokens, prompting a sequential sentence classification approach aided by a partition algorithm. Detailed elucidation of the partition algorithm is expounded upon in Chapter 5 (Discussion).

For the loss function, we deployed the unweighted CrossEntropyLoss function in PyTorch, since a dynamically weighted loss function, even though it would be more appropriate for unbalanced datasets (such as TRAM's original training dataset in figure 3.3.2), could only deliver marginal increase in terms of F1-score with BertForSequenceClassification [51]. In all our experiments, we will either train or fine-tune the SciBERT model with batches of ten sentences.

[20] https://huggingface.co/allenai/scibert_scivocab_uncased

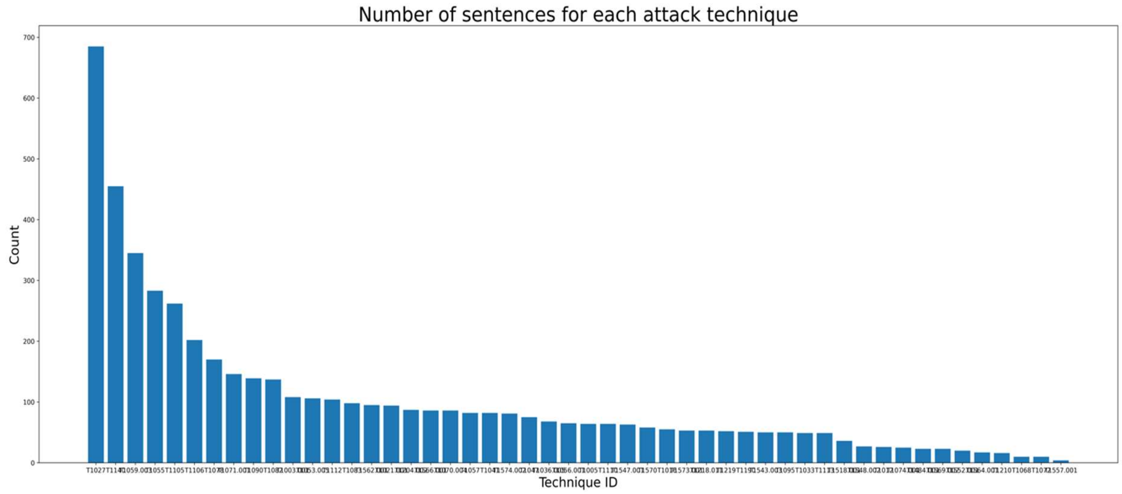

Figure 3.4.2. The dataset used for initial training of TRAM. There are severe class imbalances in this dataset: T1027 and T1059.003 have nearly 700 and over 450 sentences, respectively, while there are techniques with roughly ten sentences.

To rebalance the dataset for retraining TRAM, we used another dataset (CTI-to-MITRE with NLP) that includes sentences annotated with technique IDs from MITRE ATT&CK framework. This annotation process was manually done by researchers [12]. We have deployed multiple methods for preprocessing:

- Firstly, we would choose only sentences with techniques identifiable by TRAM.
- Then, for techniques having under 100 examples in the new dataset, we would ask GPT-3.5 to find synonyms of sentences with these techniques.
- Finally, for two techniques with the highest number of sentences (which, coincidentally are also T1027 and T1059.003), we randomly removed half of their sentences.

The result is displayed in figure 3.4.3.

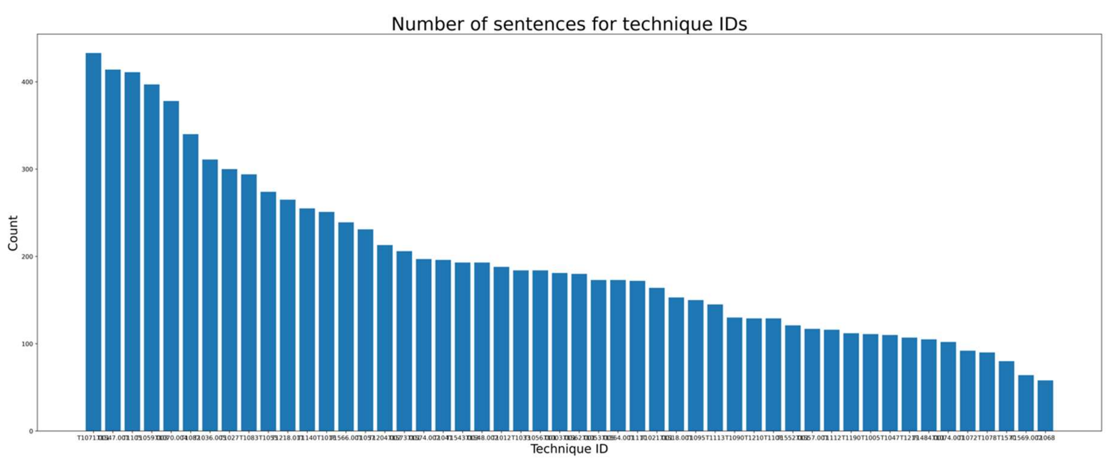

Figure 3.4.3. The structure of the new dataset, after deployment of rebalancing techniques. Though there are still imbalances between techniques in the new dataset, there are no outliers (techniques having too many or too few sentences).

In total, the rebalanced dataset contains 9811 sentences. For the training/fine-tuning tasks, we divided the dataset into training set (80% of the dataset) and test set (20%). Our training and fine-tuning processes were conducted on the training sets.

To find the best method to improve TRAM's classification performance, we conducted three tests on different configurations:

- SciBERT model retrained with the modified dataset (in 80% confidence level).
- SciBERT model fine-tuned with the modified dataset (in 80% confidence level).
- SciBERT model retrained with 5-fold validation with the modified dataset (in 80% confidence level).

These tests were carried out with an 80% confidence level because we believed that these models can at least match the initial SciBERT model provided by TRAM's authors, and thus outperform the original TRAM configuration with a 25% confidence level. Furthermore, because these models were trained or fine-tuned over six epochs (as was the initial SciBERT model), we expected there to be less of a difference in terms of hyperparameter tuning and, thus, levels of confidence.

3.4.4. Proposed approach: Extracting techniques from a summarized report by GPT-3.5, using SciBERT trained with a rebalanced dataset.

Apart from resolving class imbalance (as mentioned in part 3.4.3), there are multiple potential methods to improve further the classification performance for a ML model:

- Reducing the number of epochs, since with our in-context learning process for SciBERT, the entire model is trained with a new dataset, and with the error being back-propagated to the whole model' architecture for each training epoch [52], it would be prone to overfitting.
- Summarizing CTI reports to get sentences containing attack techniques (as mentioned in part 3.4.2), since with the verbosity of threat reports as mentioned in chapter 1, CTI extractors could mislabel sentences that are not describing cyber-attacks as describing one.

Consequently, a novel methodology for deploying TRAM was devised, encompassing the implementation of various strategies aimed at enhancing technique classification performance.

1. **Utilizing CTI Reports as Input:**

CTI reports were employed as input, and ChatGPT was tasked with summarizing these reports, employing the identical prompt delineated in Section 3.4.2. The sentences extracted by ChatGPT were subsequently compiled and stored in separate text files for each threat report.

2. **Training Process for SciBERT:**

The initial SciBERT model, pre-trained on scientific text, underwent a training process utilizing our rebalanced CTI dataset, as explicated in Section 3.4.3. The number of training epochs was curtailed to four, a decision informed by our experimental findings revealing that this epoch count represents the zenith, wherein the validation loss remains lower to the training loss, thereby mitigating the risk of overfitting (as illustrated in Figure 3.4.4). The learning rate, ε, and other training parameters remained unaltered in comparison to those detailed in Section 3.4.3.

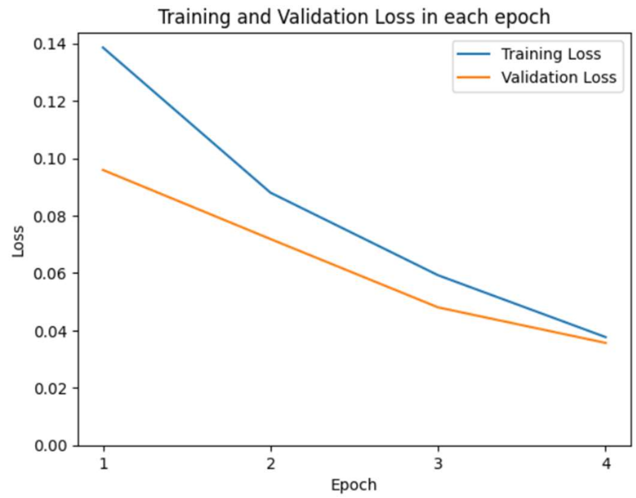

Figure 3.4.4. Average training and validation loss for each epoch in the training process for our proposed approach with SciBERT model. It is noticeable that the training and evaluation loss would be converged after epoch four.

The full pipeline for our approach (with SciBERT model trained in four epochs) is displayed in figure 3.4.5.

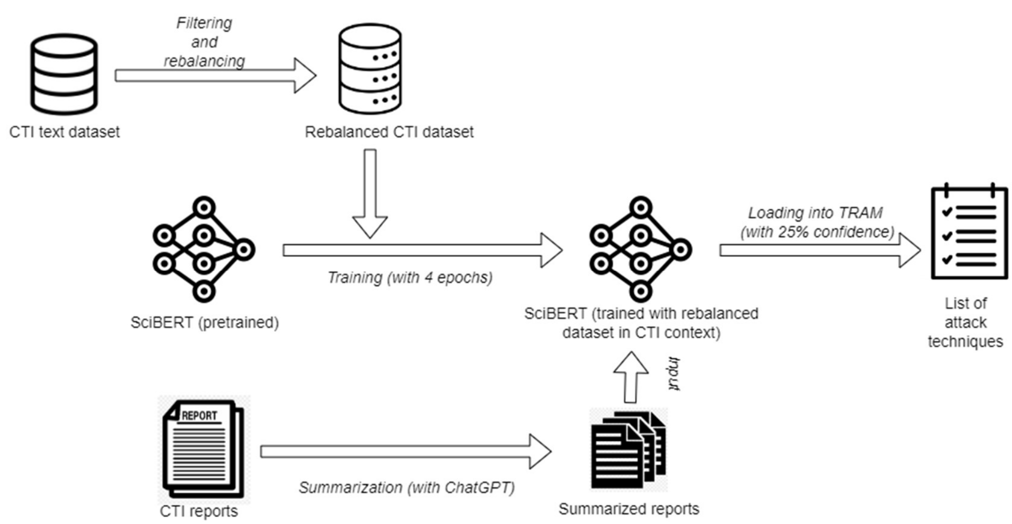

Figure 3.4.5. Our proposed attack technique extraction pipeline for TRAM (using SciBERT model trained with four epochs to extract techniques from summarized reports). It is worth noting that the approaches in our test are like this pipeline; the differences are in terms of how many epochs are used for training, as well as what is the input and confidence level).

For this model, we assessed our approach alongside loading the full CTI report into the new SciBERT model, at 25% confidence level (like how we evaluated basic SciBERT model in part 3.4.2). The reason we chose this confidence level, rather than 80% like in part 3.4.3, is that since we reduce the number of epochs, the difference in performance between our new model and the approaches in part 3.4.3 is higher (with more difference in parameters).

These tests would be mentioned alongside approaches in part 3.4.3, since they are all methods that we want to test for potential improvements against basic configurations of TRAM.

3.5. EVALUATION METRICS

To find a method that could deliver the highest performance in technique extraction for CTI reports, we will conduct quantitative analysis of CTI extraction programs. Specifically, we will evaluate the outputs generated by these programs for each CTI report and calculate classification metrics for these reports (since our research problem could be understood as how to classify text as belonging to an attack technique or not). Metrics chosen for this task are [53]:

+ Precision: $\frac{True\ Positive}{True\ Positive + Fals\ \ Positive}$
+ Recall: $\frac{True\ Positive}{True\ Positive + False\ Negative}$
+ F1-score: $2\ \times \frac{Precision\ \times Recall}{Precision + Recall}$

A good model should be able to maximize their F1-score, with the number of false positives being as low as possible (since as we mentioned in part I, one of the greatest problems with current attack detection is the high number of false alarms).

Apart from these metrics, we would also consider other data related to performance of CTI classification programs:

- Number of false positives/false negatives generated by CTI extraction models. False positives in this situation mean techniques identified by extraction models but are not presented in the report as ground truth, while false negatives are techniques presented in ground truth but not found by extraction programs.
- Number of true positives identified by extraction models (techniques presented in the ground truth and found by these programs). It is worth noting that we only analyse true positives at the report level (if the technique is presented in both the ground truth and the outputs, we consider it a true positive), and not at the sentence level (that is, we would not find that whether sentences that the model consider as a part of a technique are true or false).

Regarding the determination of true positives, the criteria are contingent upon the assessments conducted and the CTI extraction programs employed in our study:

- **Comparison between AttackG and Open-Source LLMs:**

For the evaluation involving AttackG and open-source Language Model Models (LLMs), particularly in instances where AttackG does not assess sub techniques within the MITRE ATT&CK framework, correctness is decided based on the congruence of the extracted techniques with the parent techniques associated with sub techniques in the report. Notably, if AttackG designates a result that aligns with the parent techniques corresponding to sub techniques in each report, it is deemed correct. For instance, if AttackG identifies the sub technique T1566.001 in the OilRig report (Figure 3.2.1) as T1566, then the result is considered accurate.

In the context of LLMs, mitigating the risk of hallucination associated with IDs, only the technique name is considered when determining the correctness of results. Using the aforementioned OilRig report as an illustration, if an LLM correctly identifies Phishing [54] as the attack technique, irrespective of any inaccuracies in the MITRE ATT&CK ID or the name, it is deemed correct. This leniency extends to cases where the LLM misattributes the ID but accurately identifies the technique from sub techniques, such as identifying Spearphishing (as the name could still be ascertained from sub technique [55]).

- **Evaluation of TRAM (Second and Third Tests):**

In the assessment of TRAM during our second and third tests, only techniques that exhibit correctness in both ID and names are classified as true positives. This stringent criterion stems from TRAM's classification methodology, which relies on exact matching rather than fuzzy matching akin to AttackG. Given the precise labelling and unambiguous nature of TRAM's classifications, the risk of hallucination is diminished compared to other LLMs.

It is imperative to note that due to the structure of the JSON file from the Attack-Technique-Dataset, which solely contains lists of techniques corresponding to each CTI report, the association between techniques and specific sentences remains elusive. Consequently, the determination of true positives is based solely on the presence of a technique within the list of techniques associated with a given CTI report in both datasets subjected to our testing.

Chapter 4. Evaluation results

4.1.COMPARISON BETWEEN ATTACKG AND OPEN-SOURCE LLMS

To implement the tests, we chose six reports from Adversary Emulation Library (as mentioned in part 3.1), since these reports are short (under 500 words); thus, this would solve two main problems with AttackG:

- Guaranteeing the runtime to process CTI report being short (our test environment does not have a Graphical Processing Unit (GPU), as a personal laptop; thus, for reports longer than 500 words, AttackG might take over 2 minutes to build a knowledge graph).

- Preventing dependency explosion, which as explained in part 2.3, is a problem happening frequently when a long CTI report is processed.

The results are displayed in tables 1, 2 and 3.

Table 1*: Comparison of AttackG and versions of Llama 2 in terms of false positives and false negatives.* (Note that false negatives are displayed with a minus "- "sign, and false positives are displayed with a plus "+" sign).

Report	*Number of techniques in ground truth*	*Number of false negatives/ positives (AttackG)*	*Number of false negatives /positives (Llama-2-7b-chat)*	*Number of false negatives/ positives (Llama-2-13b-chat)*	*Number of false negatives/positives (Llama-2-70b-chat)*
APT29	3	(-3) + 9	(-2) +9	(-2) +10	(-0) +9
Carbanak	6	(-6) + 3	(-1) +5	(-5) +8	(-3) +8
FIN6	24	(-24) +3	(-21) +7	(-24) +8	(-22) +19
FIN7	4	(-4) +7	(-4) +7	(-3) +14	(-2) +7
menuPass	5	(-3) +19	(-4) +5	(-4) +11	(-5) +12
OilRig	4	(-2) +9	(-1) +5	(-2) +5	(-2) +8

Table 2: *Comparison of AttackG and versions of Llama 2 in terms of true positives.*

Report	*Number of true positives (AttackG)*	*Number of true positives (Llama-2-7b-chat)*	*Number of true positives (Llama-2-13b-chat)*	*Number of true positives (Llama-2-13b-chat)*
APT29	0	1	0	3
Carbanak	0	5	1	3
FIN6	0	3	0	2
FIN7	0	0	1	2
menuPass	2	1	1	0
OilRig	2	3	2	2

Table 3: *Average precision, recall and F1-score of AttackG and versions of Llama2.*

	AttackG	*Llama2-7b-chat*	*Llama2-13b-chat*	*Llama2-70b-chat*
Average precision	0.0416	0.2403	0.0911	0.1733
Average recall	0.15	0.3736	0.2792	0.4306
Average F1-score	0.0701	0.2733	0.1199	0.2384

The observed classification performance of AttackG reveals a pronounced deficiency, notably evidenced by its notably low true positive rate, as evidenced by four out of six reports yielding no true positives. Moreover, the comparative analysis indicates that AttackG performs even worse than certain versions of Llama 2 that have not undergone fine-tuning. A noteworthy observation is the substantial number of false positives generated by AttackG, exemplified by the most extreme case, menuPass, wherein the number of false positives is approximately four times greater than the actual count of techniques.

The primary factor contributing to the suboptimal performance of AttackG is attributed to its reliance on a template-matching based method, wherein predictions of techniques are derived from historical examples. This methodology proves to be inherently unsuitable for discerning extensive sections within lengthy documents, as it may indiscriminately identify a document segment as conforming to a technique template. Consequently, this predisposition results in a higher ratio of false positives [5].

A secondary factor influencing the performance relates to AttackG's approach to identifying technique templates, which involves crawling past reports from the MITRE ATT&CK database. The inherent variability in the number of reports for each technique introduces bias into AttackG's results, favouring techniques with a higher prevalence in the MITRE ATT&CK database, as illustrated in Figure 4.1.1. This aligns with the findings of [56], where the authors highlighted the diminished applicability of fuzzy-based classification in sentiment analysis on Twitter due to the disparate distribution of datasets.

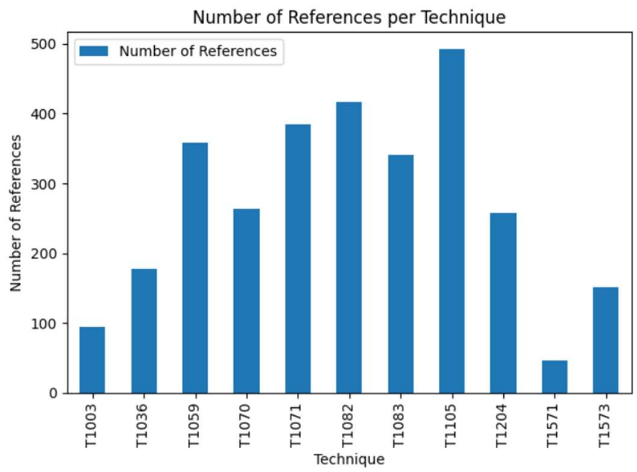

Figure 4.1.1. Number of references for each technique that AttackG found from the MITRE ATT&CK database, being false positives at least twice. Given that AttackG's authors based their work on 7373 references (technique templates) [4], these techniques have a significant proportion in the database and could influence the bias of fuzzy matching algorithms towards them.

An additional noteworthy consideration pertains to the fuzzy matching methodology employed by AttackG, which hinges on approximating the Levenshtein distance between each word and the technique template. Significantly, each execution of the program yields different distance values [57], rendering this matching method non-deterministic. Consequently, for the same CTI report, multiple runs of AttackG may generate disparate lists of techniques. A pertinent illustration of this issue is evident in the examination of the OilRig Campaign report [41], wherein the initial test yielded 11 techniques (as depicted in Table 1 and 2), whereas a subsequent analysis revealed the identification of 24 distinct techniques from the same report.

Concerning the contemporary iterations of Llama 2, the observed suboptimal classification performance aligns with expectations, given that these models were trained on a broad corpus of knowledge rather than being specifically tailored to threat intelligence data. It is imperative to underscore that when querying attack techniques alongside their IDs, Llama 2 chatbots exhibit a tendency for hallucination. This manifests either in the fabrication of a technique ID or in the confusion between technique and tactic IDs within the MITRE ATT&CK database, as exemplified in Figure 4.1.2. Consequently, it is necessary for users to possess prior knowledge about the MITRE ATT&CK framework to accurately discern the actual technique name before mapping the result to the respective CTI report.

1. Spearphishing: Carbanak uses spearphishing emails to trick victims into downloading malicious documents or clicking on links. This technique is identified as TA0001 - Phishing in the MITRE ATT&CK framework.
2. Malware: Carbanak uses custom malware, specifically the Anunak malware, to gain access to the targeted systems. This technique is identified as TA0002 - Malware in the MITRE ATT&CK framework.
3. Certificate Abuse: Carbanak abuses digital certificates to make their malware appear legitimate. This technique is identified as TA0005 - Certificate Abuse in the MITRE ATT&CK framework.
4. Command and Control: Carbanak establishes command and control (C2) channels with their C2 servers using malicious files embedded in Microsoft Word documents. This technique is identified as TA0006 - Command and Control in the MITRE ATT&CK framework.
5. Lateral Movement: Carbanak moves laterally across networks to find critical systems that manage financial transactions. This technique is identified as TA0008 - Lateral Movement in the MITRE ATT&CK framework.
6. Privilege Escalation: Carbanak escalates privileges to gain access to sensitive data and systems. This technique is identified as TA0010 - Privilege Escalation in the MITRE ATT&CK framework.
7. Persistence: Carbanak establishes persistence on compromised systems using Windows native tools, such as scheduled tasks and auto-run services, or other non-malicious tools, such as VNC. This technique is identified as TA0011 - Persistence in the MITRE ATT&CK framework.
8. Data Encryption: Carbanak encrypts data to evade detection and make it more difficult to recover. This technique is identified as TA0012 - Data Encryption in the MITRE ATT&CK framework.
9. Anti-Forensic Techniques: Carbanak uses anti-forensic techniques, such as deleting logs and wiping evidence, to cover their tracks. This technique is

Figure 4.1.2. A part of the result when prompting Llama-2-13b-chat to ask about the techniques and corresponding IDs in the Carbanak report, from [40]. The chatbot frequently considers tactics (lateral movement, privilege escalation, persistence, command and control) as techniques in MITRE ATT&CK framework. Furthermore, the model also hallucinates when generating the corresponding IDs, since TA are initials for tactics in MITRE ATT&CK database; and even their tactics ID do not correspond to the actual tactics [58]. It should be noted that even though Llama-2-13b-chat could detect an exact technique (Spearphishing), however, since this name could be identified with the actual technique (Phishing) as we mentioned from part 3.3.1, we accepted the output from the chatbot.

As for h2oai/h2ogpt-gm-oasst1-en-2048-falcon-7b-v2 model, it could not find any attack technique at all; instead, the chatbot returned a generic response (this is attached in Appendix B). This is due to the fact that, after being trained on a wide corpus of knowledge such as Llama 2, the chatbot was further refined using the oass1 dataset from OpenAssistant [59] , which is devoid of any information pertaining to the MITRE ATT&CK framework [59]. This forced the old weight in the Falcon-7b model to be overwritten (in a process called “catastrophic forgetting” [60]), and led to the model’ inability to extract attack techniques from a CTI report.

In summary, it is evident that non-fine-tuned Large Language Models (LLMs) exhibit suboptimal performance in the identification of attack techniques. However, AttackG, distinguished by its probabilistic matching model, performed even more poorly. This can be attributed to the inherent limitations of AttackG's approach, rendering its outputs unreliable for the identification of attack techniques. Consequently, the non-fine-tuned version of LLMs cannot be leveraged to enhance the performance of AttackG, as the resultant outputs would introduce a high false positive rate, thereby confounding analysts rather than augmenting their capabilities.

4.2.COMPARISON BETWEEN DIFFERENT APPROACHES FOR TRAM'S DEPLOYMENT.

For TRAM analysis, we would test the results in both datasets for 10 reports. As mentioned in section 3.2.2, there would be four tests being conducted for this comparison. Due to higher number of graphs and test cases compared to our previous tests, we would focus on precision, recall and F1-score of these approaches.

The results are displayed in figures 4.2.1, 4.2.2 and 4.2.3.

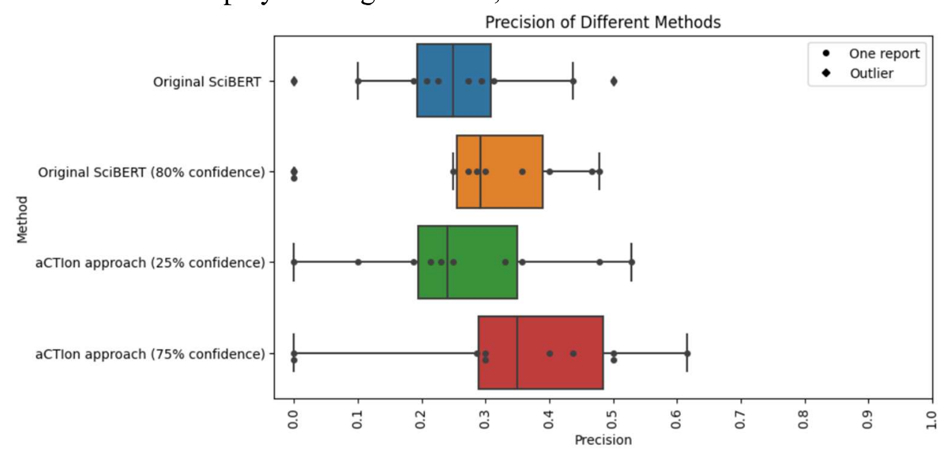

Figure 4.2.1: Precision for different configurations of TRAM.

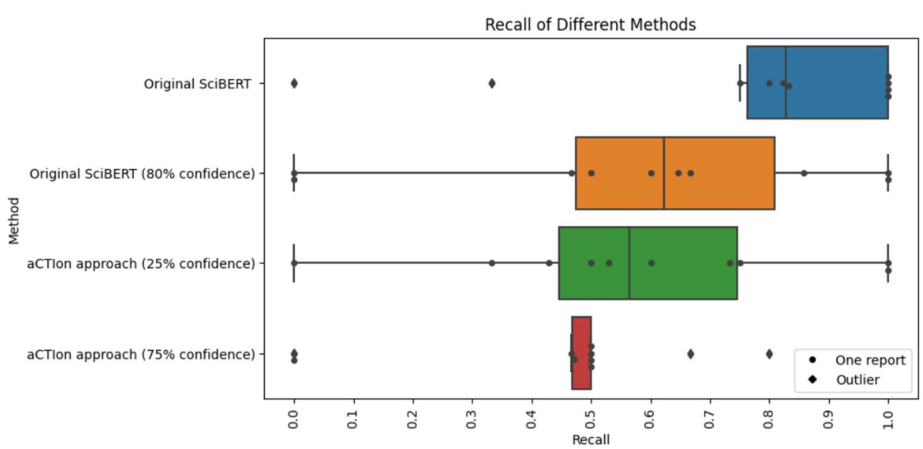

Figure 4.2.2: Recall for different configurations of TRAM.

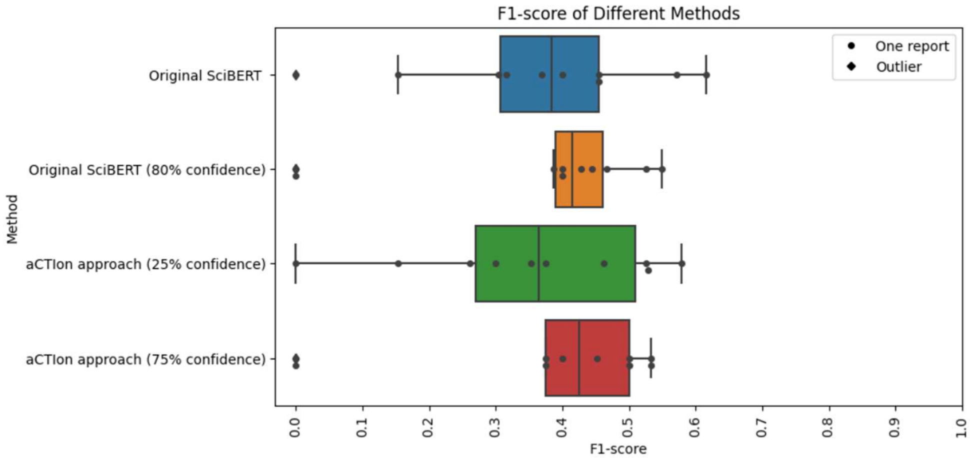

Figure 4.2.3. F1-score for different configurations of TRAM.

These figures show that the original TRAM website, even though having a high recall, has a low performance in terms of F1-score (roughly 0.40), due to low level of precision rate. This is because in the SciBERT model used for the website was trained on six epochs in the original pipeline [21], and through our observation of training and validation loss, it shows that after 4 epochs, the model begins to be overfitted (with higher validation loss than training loss), in figure 4.2.4:

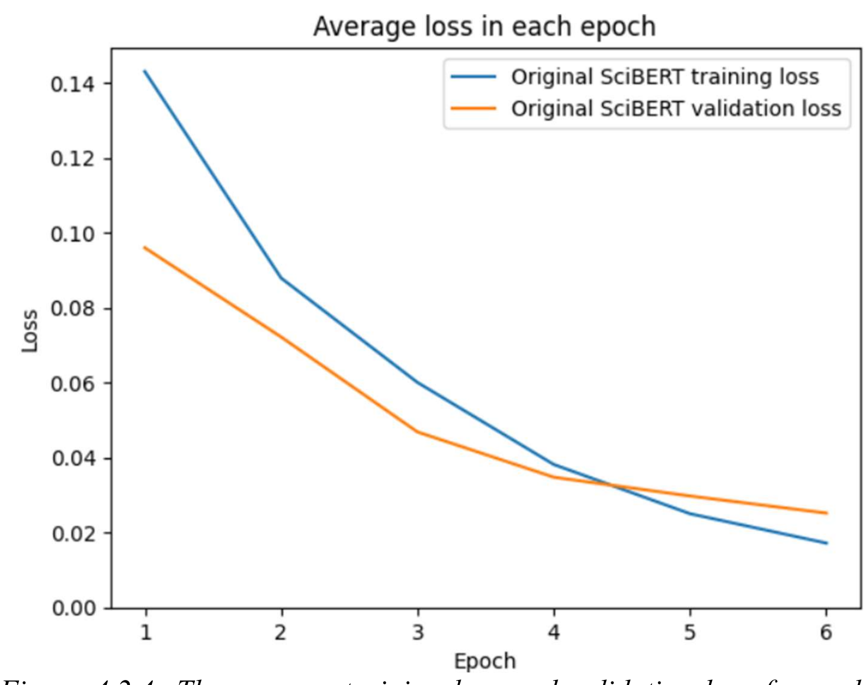

Figure 4.2.4: The average training loss and validation loss for each epoch when training SciBERT with the original pipeline.

The figures presented earlier demonstrate that elevating the confidence level for TRAM to 80% can enhance the precision rate, consequently improving the F1-score

[21] https://github.com/center-for-threat-informed-defense/tram/blob/main/model-development/train_single_label.ipynb

by eliminating a substantial number of techniques. However, it is imperative to acknowledge that, given the pre-existing overfitting of the model [61] , the confidence level for the selected techniques remains high. Consequently, while this adjustment augments classification performance, the enhancement may not be of sufficient magnitude to warrant the integration of TRAM into the workflow of analysts.

In the context of aCTIon's summarization approach, there is a discernible reduction in the volume of text input into TRAM, as illustrated in Figure 5.2.5. Nevertheless, it is noteworthy that TRAM, in its quest to identify text containing attack techniques, subjects the report to a dual summarization process. This dual summarization contributes to a diminished recall, elucidating the rationale behind the comparatively lower F1-score observed in the aCTIon approach when employing the original report at a 25% confidence level (as delineated in Figure 4.2.3). Only by elevating the confidence level to 75% does TRAM succeed in mitigating false positives sufficiently to surpass other configurations in terms of F1-score.

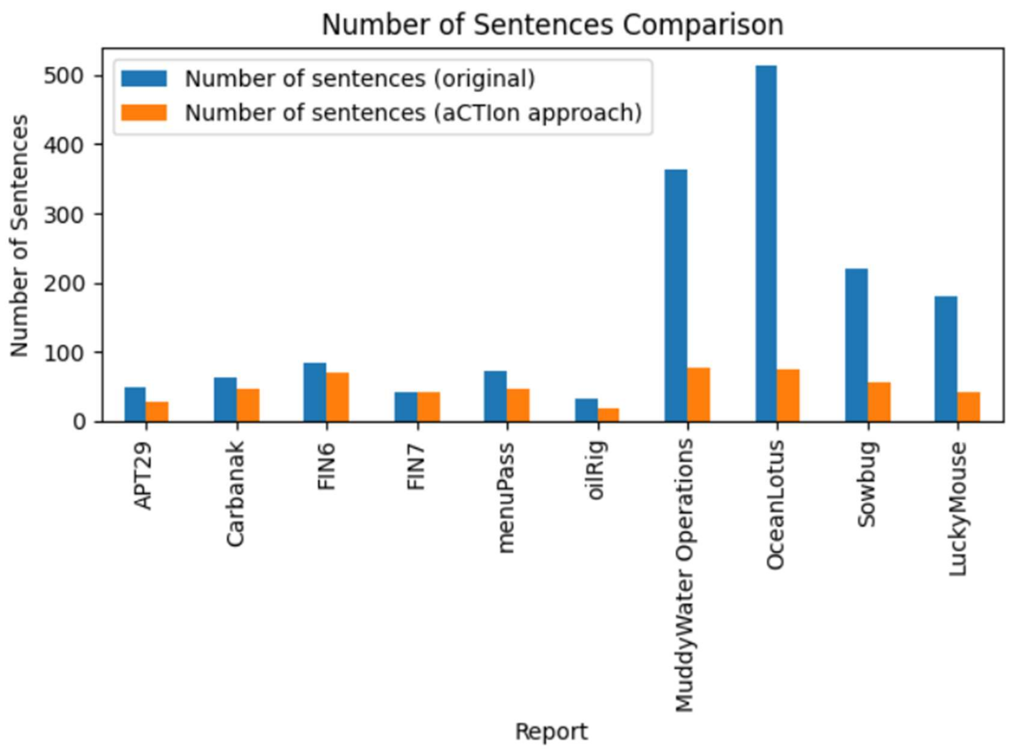

Figure 4.2.5. The number of sentences of each report, before and after using GPT-3.5 to find sentences containing attack techniques. Please note that due to a sentence partitioning algorithm that TRAM uses for processing text, each sentence in original report would be partitioned into multiple parts; thus, the number of sentences here are inflated compared to the actual number of sentences in the dataset. We would discuss more about the partitioning algorithm in the Discussion part.

4.3.COMPARISON BETWEEN TECHNIQUES TO IMPROVE TRAM'S PERFORMANCE.

As demonstrated in Section 3.2.3, the scope of this comparison entails the evaluation of five distinct techniques across a set of ten reports. Given the substantial volume of test cases for each report, our primary emphasis in the assessment lies on precision, recall, and F1-score. It is imperative to highlight that the benchmark for this comparative analysis is the original SciBERT model configured with a 25% confidence level. This benchmark serves as a reference point, enabling us to gauge the capacity of each model to surpass the classification performance achieved by the original configuration of TRAM.

The results are displayed in figures 4.3.1, 4.3.2 and 4.3.3.

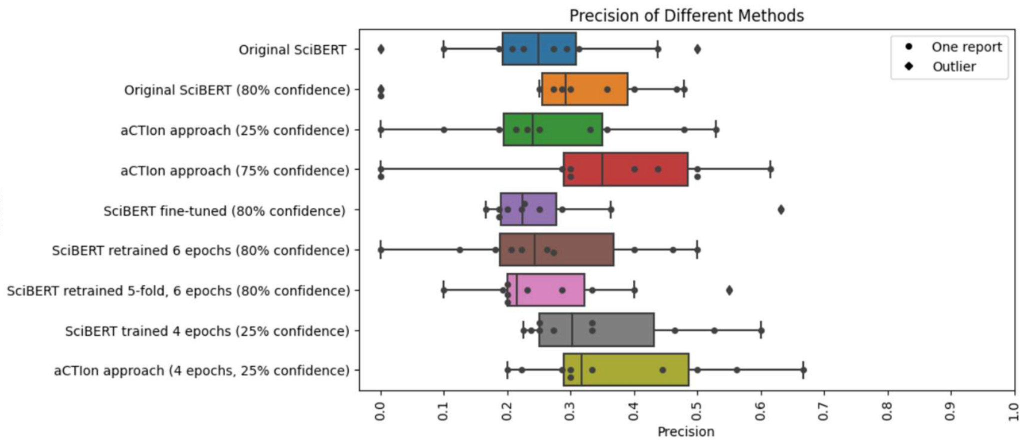

Figure 4.3.1. The precision of our methods for evaluating TRAM.

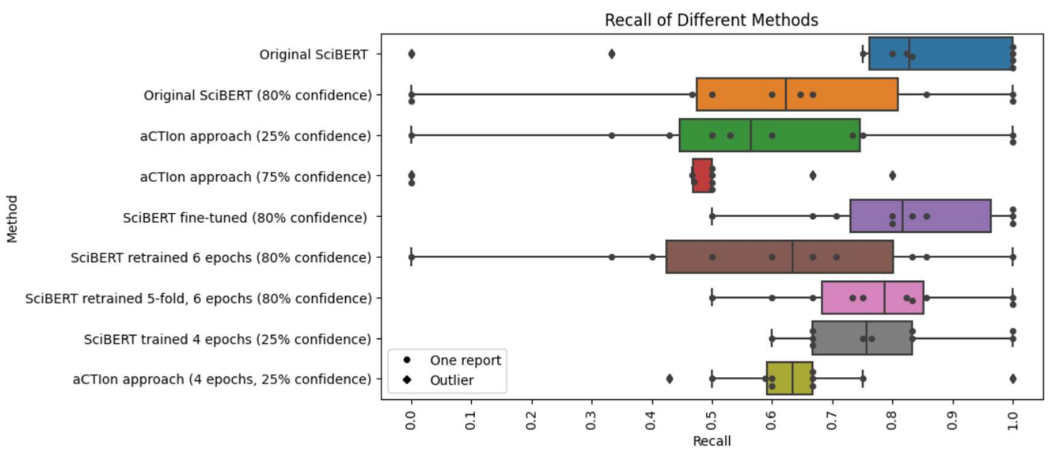

Figure 4.3.2. The recall of our methods for evaluating TRAM.

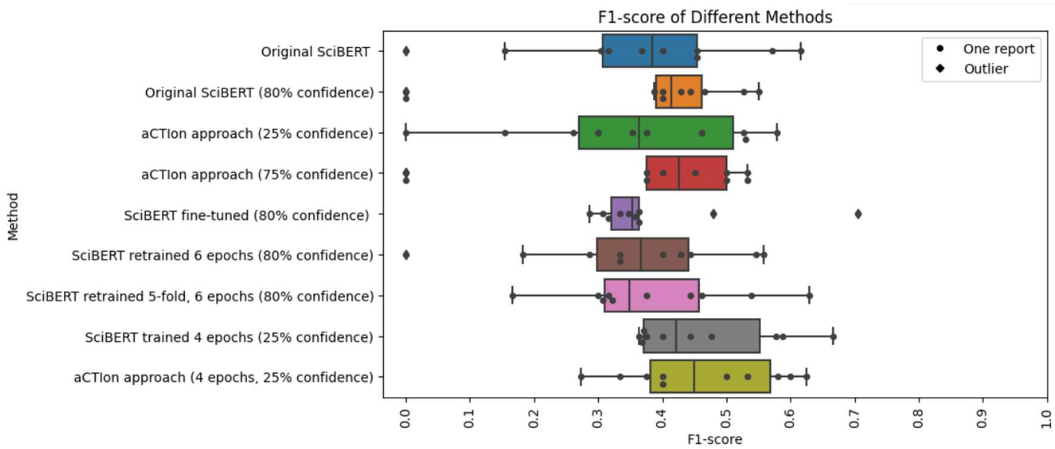

Figure 4.3.3. The F1-score of our methods for evaluating TRAM.

The performance analysis depicted in Figure 4.3.3 reveals that SciBERT models, whether trained or fine-tuned over six epochs, failed to yield any improvement in the F1-score compared to the original TRAM configuration. Strikingly, these models even exhibited lower F1-scores when contrasted with the original SciBERT models configured with an equivalent confidence level (80%). This downturn in performance can be attributed to the commonality in their training and fine-tuning process, which was conducted over six epochs, consequently succumbing to the overfitting issues detailed in Section 4.2.

This overfitting phenomenon is demonstrated further through the representation of the training loss dynamics, as demonstrated in Figure 4.3.4.

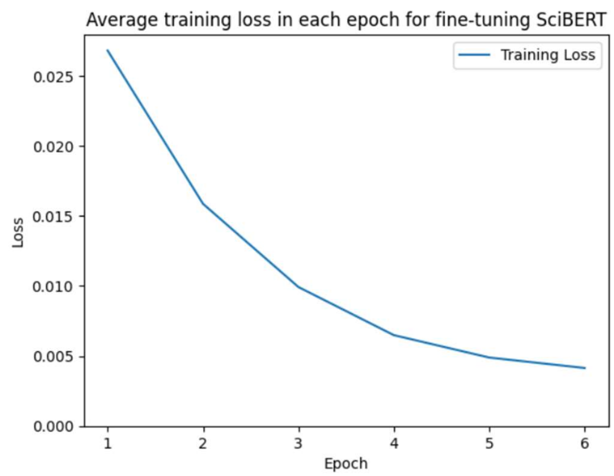

Figure 4.3.4: The average loss for fine-tuning SciBERT in each epoch. The loss curve on fine-tuning is like what happened in the original training process in part 5.2: from the fourth epoch, the loss would begin to stabilise. This indicates that using more than four epochs to fine-tune the models would increase risks of overfitting.

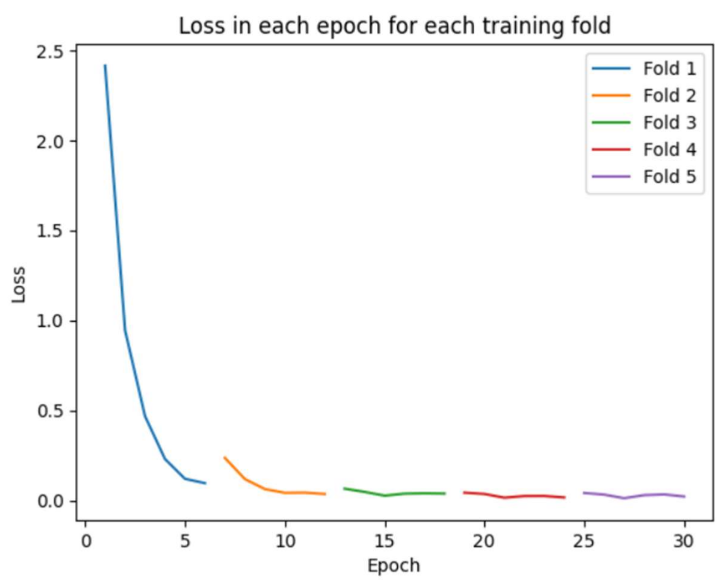

Figure 4.3.5: The loss in each epoch for using 5-fold validation. Since we do not split the dataset into training and test sets for this method, we need to use 5-fold validation

to get the same training and test results; however, this raises the number of epochs to 30 overall, and as mentioned in part 5.2, only 4 epochs are enough to train SciBERT without overfitting. Furthermore, the dataset in the original model has already been stratified, so using 5-fold validation is unnecessary.

Furthermore, in the training process, the SciBERT model has already remembered the weight of each technique in the dataset (since the dataset was already stratified by technique). Therefore, with a new dataset being used for fine-tuning, the weight of techniques in the pre-trained model would be overwritten (similar to the 'catastrophic forgetting' problem in the case of h2oai/h2ogpt-gm-oasst1-en-2048-falcon-7b-v2 model in part 4.1), leading to a decline in performance. The only upside from using the rebalanced dataset (for all of these models) is that for shorter reports, there are a higher chance of getting a true positive, since techniques with lower number of references, such as T1190, T1072 or T1484 would be detected more with a rebalanced dataset (only the retrained model with 6 epochs has no true positive, indicated by zero precision and recall in figures 4.3.1 and 4.3.2).

It became evident that a paramount strategy for enhancing SciBERT's performance is the reduction of the number of training epochs, with subsequent attention to rebalancing classes in the training dataset, to address the challenge of overfitting in our models. Consequently, our final iteration of training involved leveraging the rebalanced dataset, limiting the training process to four epochs. This modification yielded a substantial improvement, reflected in a median F1-score increase of approximately seven percentage points, thereby surpassing the original baseline that utilized the unaltered SciBERT model without report summarization.

The efficacy of this approach is underscored by its ability to detect true positives across all reports, demonstrating a nuanced understanding of techniques that were underrepresented in the original dataset. Unlike the original SciBERT model, irregularities such as zero precision or recall were absent in this enhanced model.

However, it is essential to acknowledge the inherent trade-off associated with this training methodology. Specifically, techniques characterized by a limited number of sentences may experience a decline in classification performance, as exemplified in Table 4. The test set reveals several techniques that the trained SciBERT model fails to identify any true positives, including T1068 and T1569.002. This explains why our new model could not improve the F1-score substantially; in figure 4.3.3, even after using summarisation approaches from aCTIon, the median F1-score of our new model (with 4 epochs) is only approaching 0.45 (with 25% confidence), which is only 4 percentage points higher than using the same summarisation method for the original SciBERT model. Even so, with the outperformance compared to the original baseline, this method could be considered a reliable alternative for analysing techniques from CTI reports.

Another point worth noting is that since we have reduced the number of epochs to prevent overfitting, the confidence level of our proposed SciBERT model could be decreased, which is consistent with current theory [61]. This means that choosing an optimal confidence level (to maximize F1-score) would be a priority for further research, since we have not been able to find an optimal level for confidence due to time constraints.

Table 4: *Classification results for the test set (20% of the training dataset), for the SciBERT model trained with four epochs with the rebalanced dataset.*

Technique ID	*Precision*	*Recall*	*F1-score*	*Number of sentences in the test set*
T1056.001	0.800000	0.864865	0.831169	37.0
T1057	0.793103	1.000000	0.884615	46.0
T1059.003	0.684783	0.797468	0.736842	79.0
T1068	0.000000	0.000000	0.000000	12.0
T1070.004	0.900000	0.947368	0.923077	76.0
T1566.001	0.867925	0.958333	0.910891	48.0
T1569.002	0.000000	0.000000	0.000000	13.0
T1570	1.000000	0.312500	0.476190	16.0

Chapter 5. Discussion

5.1. WHICH METHOD COULD EXTRACT ATTACK TECHNIQUES FROM THE MITRE ATT&CK FRAMEWORK MOST EFFECTIVELY?

From our analysis in part V, we could identify two separate groups of methods that could be used to extract attack techniques:

- **Knowledge graphs** (such as AttackG and Extractor): The potential main advantage for applying knowledge graph, in this case, is to make the information easier to convey for cyber analysts. However, even though progress has been made to classify various types of text (apart from IOCs) to different entity types (such as systems, files, or vulnerabilities), mislabelling could still happen for entities in a CTI report, with the graph for OilRig campaign in figure 3.2.1 as an example (where OilRig is identified as two different entities: executable and file, despite the definition of OilRig as a threat actor, which should be represented as a circle). Another problem of AttackG is that the graph does not contain any information about which sentence does each edge belong to, which make the reference of information to the report difficult for cyber analysts.

  Another knowledge graph builder (Extractor [7]) has also been tested in terms of entity extraction. However, neither the retrained BERT model nor the dataset used for retraining, as its authors mentioned in the paper, have been provided, although Extractor is an open-source program. As such, both the BERT model and the Semantic Role Labelling models that the authors mentioned in the paper was not trained with CTI-related texts. As a result, even if the program provides verbs in the graph to assist users in recognizing events occurring during a cyber incident, the graphs produced by Extractor remain disorganized (as illustrated in figure 5.1.1), rendering them unusable for identifying attack strategies.

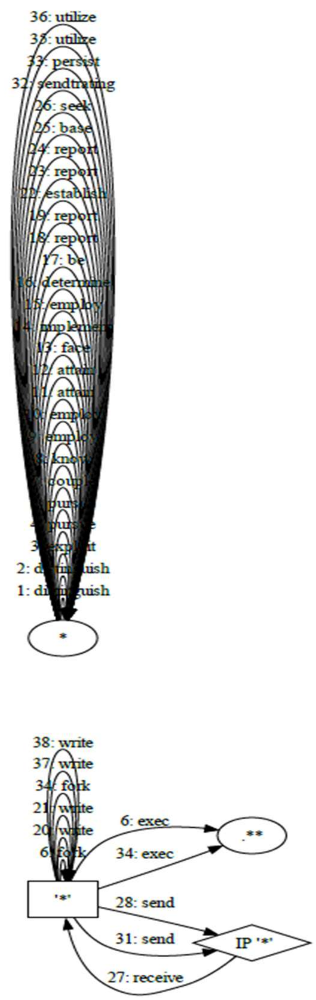

Figure 5.1.1: Output generated from Extractor for the CTI report of the APT29 campaign [62]. Since the new model could not recognize entities from the threat report, each of them is represented as an asterisk (meaning 'anything' for Extractor). Also, without a trained model, the program would create multiple self-loops, defeating the purpose of identifying relationships and chain of actions.

Also, as demonstrated in part 4.1, AttackG, the only knowledge graph creator that identifies techniques as we know currently, has an abysmal performance in identifying attack techniques. Additionally, another problem that knowledge graph creators have not managed to fully solve is *dependency explosion* (in this case, this could happen with large threat reports), which reduce the performance of these programs by increase processing time and complexity of the report.

As such, we could conclude that these knowledge graph builders are not effective to find attack techniques from threat reports.

- **TRAM** (for programs classifying techniques by sentences): The main advantage for TRAM in technique identification is that the classification performance is higher than AttackG, though its precision and F1-score is still relatively low. However, analysts could get more accurate result by numerous methods, as described in parts 5.2 and 5.3.

  As in part 4.3, there are two methods to deploy TRAM with highest classification performance:

  + Summarizing the CTI report with GPT-3.5, and then running the program with an 80% confidence threshold.
  + Summarizing the CTI report with GPT-3.5, and then loading the retrained SciBERT model with a 25% confidence threshold.

  Although the first method has a slightly higher median F1-score due to a higher precision level, its performance was dragged down due to lower recall, especially for short threat reports (it could not detect any true positives from 2 CTI reports for APT29 and FIN7 campaigns, which is demonstrated by zero precision and recall in figures 5.3.2 and 5.3.3). In contrast, with the second method, the distribution of the F1-score is more balanced, since even techniques with low frequency in the MITRE ATT&CK database could be detected by the retrained SciBERT model.

  Furthermore, with the longer reports from Attack-Technique-Dataset (Sowbug, MuddyWater Operations, LuckyMouse and OceanLotus), we could potentially reduce the number of false positives further by increasing the confidence threshold more, and since the confidence threshold in this case is only 25% (compared to 80% in the first method), there is more room for us to boost the model performance this way.

  Another advantage of using summarized models in TRAM is that we could significantly reduce the time taken to load a report and extract techniques, especially with longer reports (such as MuddyWater Operations with sixteen pages), since the processing time for threat reports is positively correlated with their lengths. This is demonstrated in figure 5.1.2.

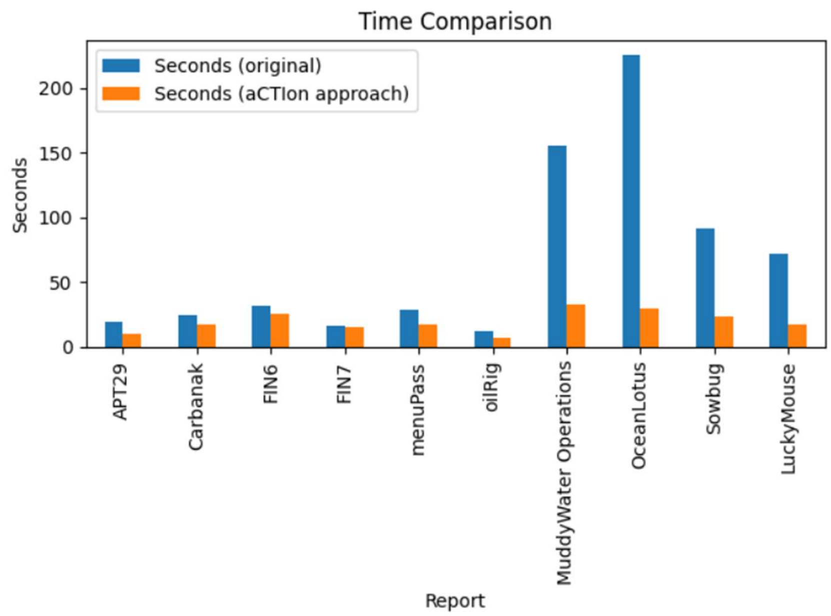

Figure 5.1.2: The runtime comparison for TRAM between using original report and using aCTIon approach for summarization.

Since the runtime is positively correlated with the length of the report (in figure 4.2.5), using summarised version of the report would drastically cut the processing time for TRAM, especially with four longer reports. This is a further benefit of summarizing reports before loading to TRAM. It should be noted that, the processing time for TRAM is not dependent on the method used to train SciBERT (fine-tuning pretrained model, training with k-fold validation or not). Rather, it depends only on the length of the threat report.

However, it should be noted that both SciBERT models for TRAM (original and trained with rebalanced dataset) were trained with only 50 most popular MITRE ATT&CK techniques, out of over 600 techniques and sub techniques [11]. As a result, the generated solutions from TRAM may miss lesser-known techniques and be less representative for the cyber threat landscape, such as with the case of APT29 report, where supply-chain compromise (T1195) was missing [63].

To summarize, TRAM's technique identification method has been more successful in finding attack techniques, and with further summarisation of CTI reports as well as using a rebalanced model, analysts could correctly identify attack techniques with lowest risk of false alarms. Therefore, the best method to extract attack techniques would be to ask GPT-3.5 to identify sentences relevant to attack techniques, then deploy the SciBERT model retrained with four epochs.

5.2. WHAT TEXT COULD BE POTENTIALLY MISLABELLED BY THREAT EXTRACTORS AS ATTACK TECHNIQUES?

Since recommendations and solutions contain words to describe attack techniques that analysts defend against; machine learning models could mislabel these parts as actual attack techniques. An example is for the LADDER program, when it is requested to find techniques from the MuddyWater Operations report, recommendations were classified as techniques like below:

```
T1211 (0.53010593354702, 'The best way to prevent attackers from finding and leveraging security holes, is to eliminate the holes altogether, including those related to improper system configurations or errors in proprietary applications.') Exploitation for Defense Evasion
T1055 (0.46477851271629333, 'An allowlisting solution to prevent certain process child-parent execution hierarchies.') Process Injection
T1534 (0.5910698026418686, 'Educate generic staff to be able to distinguish malicious behavior like phishing links.') Internal Spearphishing
T1597 (0.4832237511873245, 'Provide security staff with access to the latest threat intelligence data, which will arm them with helpful tools for targeted attack prevention and discovery, such as indicators of compromise and YARA rules.') Search Closed Sources
```

To test whether the classification performance could be further improved by removing solutions and conclusions parts, we performed these tasks with reports containing these parts (MuddyWater Operations, OceanLotus and LuckyMouse). The result was that the number of techniques (and false positives) identified decreased for the SciBERT model, trained with a rebalanced dataset, compared with using a full report (as shown in figure 5.2.1):

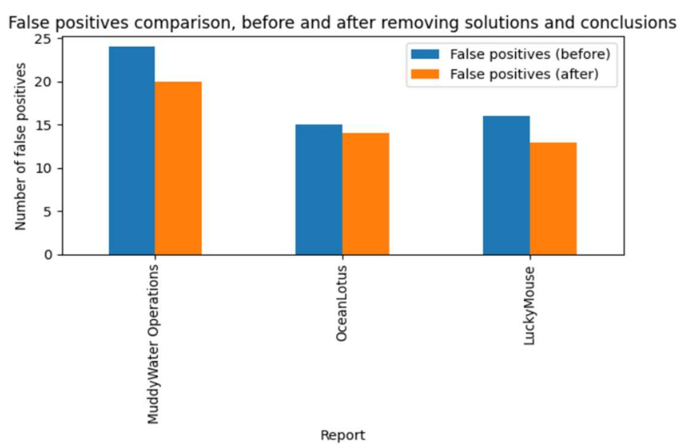

Figure 5.2.1. The comparison of false positives, before and after removing solutions and conclusions parts.

It should also be noted that, though some false positives were removed, the number of false positives remained high, and as such, the F1-score of SciBERT for these reports only increased marginally, as shown in figure 5.2.2:

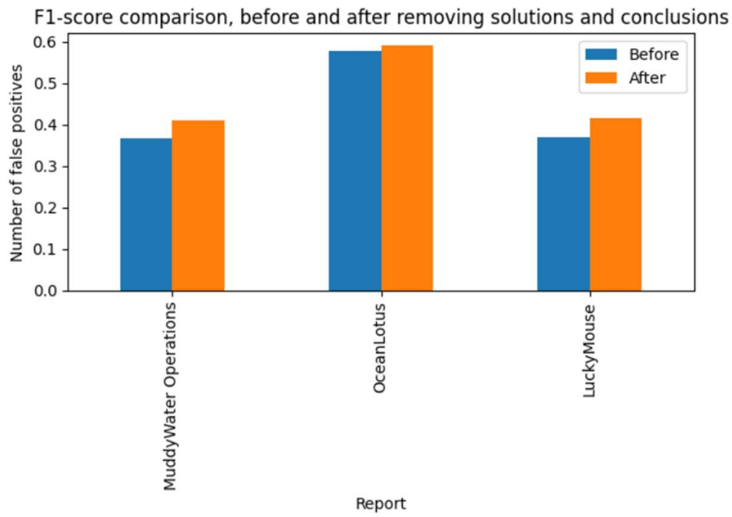

Figure 5.2.2: The comparison of F1-score for our retrained SciBERT model in three reports, before and after removing suggested solutions and conclusions.

From these findings, it could be said that the main problem for the extraction of attack techniques is that the description of the cyber attack in a report is already verbose, even with non-related parts (IOCs, recommendations, and conclusions) being removed. As such, further summarisation techniques (either by using LLMs or based on cross-validation from technical experts) should be implemented to reduce the number of false alarms and increase the classification performance.

5.3.WHAT WOULD BE THE POTENTIAL FOR ANALYSTS TO COLLABORATE WITH AI SYSTEMS FOR IDENTIFYING ATTACK TECHNIQUES?

As discussed in part 2.3, a main limitation of current CTI extractors is that there is no method for analysts to modify the content generated by Artificial Intelligence (AI). From our analysis of current knowledge graph builders (AttackG and Extractor), this problem remains acute in these programs, with no way to modify extracted entities and techniques. Given the low classification performance and high number of false positives (especially with AttackG), deferring the technique classification for human experts or with help of other AI systems is not a solution; especially with a similarly high false positive rate by current chatbots, such as Llama 2 versions (as demonstrated in part 5.1).

As a newer program, TRAM has been able to mitigate this problem by introducing a system for users to reassess their classification result. A table with sentences identified in the CTI report and their labels are presented for users; and they could choose to approve the results generated by TRAM, add new techniques into the report, or change existing results to new techniques. They could change the results based on the list of attack techniques from the MITRE ATT&CK framework, as well as software and attack groups, in case analysts want to share additional information about the attack for other teams (as shown in Figure 5.3.1). The report could then be approved by analysts before being exported to JSON or .docx formats.

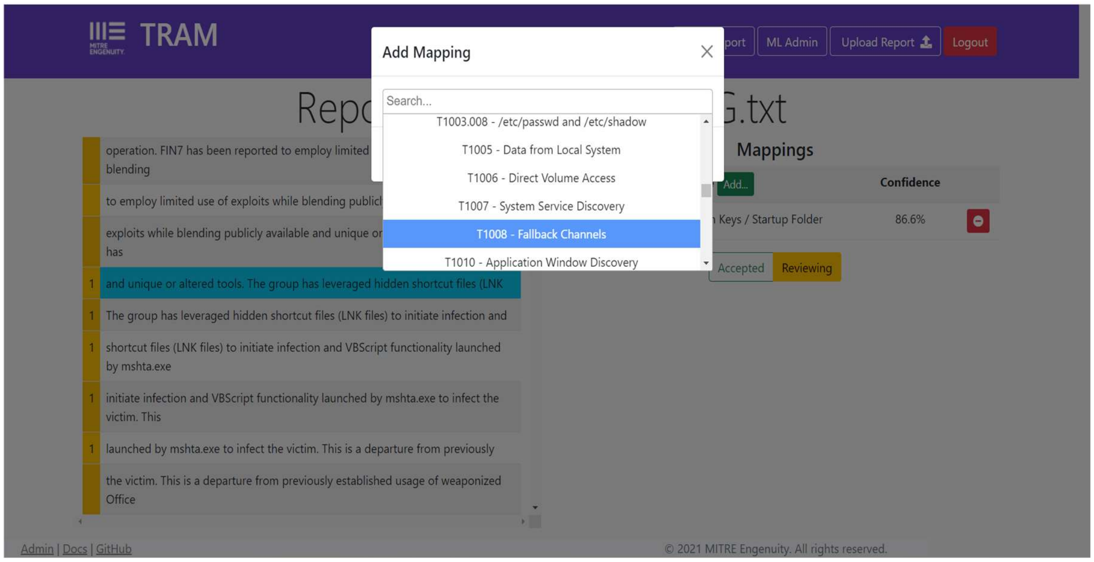

Figure 5.3.1. Options provided by TRAM to accept, modify or remove the label of a sentence in the CTI report (for FIN7 campaign [64] in this case). It is noticeable that techniques that are not in the training process for SciBERT model [11] (such as T1006 – Direct Volume Access, or T1007 – System Software Discovery) are presented in the dropdown list for users to choose from.

However, it should be noted that even though the program provided classification confidence for users in each technique, there is no method to rank the technique (or sentences) based on confidence level. This is a major disadvantage for the collaboration mechanism in TRAM, since currently, knowing the confidence of the AI model on attack techniques is important for users to prioritize techniques with low level of confidence for removal or readjustment. This is the basis for the most well-known collaboration method: learning to defer [65]. Nevertheless, TRAM's authors have confirmed that this issue would be investigated in the next round of development, therefore we would expect that more features for human-AI collaboration purpose would be available in the future [66].

Furthermore, this collaboration mechanism could work well with short reports (from Adversary Emulation Library for example, with 423 words as the highest). However, for longer reports such as OceanLotus (from figure 5.2.5), analysing all parts of sentences to make the final threat report would be a time-consuming process. As such, it is advised that analysts should summarize long reports with GPT-3.5 (like aCTIon) to simplify their collaboration steps; even though they could get higher false negatives due to more sentences being filtered out.

In summary, even though the collaboration mechanism in TRAM still has various disadvantages, it is the first mechanism (in our knowledge) that is deployed in any technique extractors, and it has potential applications to improve the classification performance. As such, this could be considered another advantage of using TRAM over other CTI extraction programs for finding attack techniques.

5.4. HOW COULD CTI EXTRACTORS FIND TECHNIQUES IN SENTENCES CONTAINING MULTIPLE OF THEM?

An important step for technique classification is to identify sentences, before applying machine learning models to predict techniques, as demonstrated by LADDER's authors [5]. In the case of TRAM, its authors, rather than breaking the report by sentences, deploying a window sliding techniques to partition the report:

1. The first thirteen words would be extracted from the CTI report.
2. For each five words in a report, another thirteen words would be loaded into the TRAM program.
3. The process would continue until the end of the report (as such, there could be some cases where the last several words from the end of the report could be loaded).

Then, these parts of the sentences would be loaded into the SciBERT model (trained with a single label for each sentence). The process could be described by this pseudocode:

```
function ReportPartitionToSentences(report):
    lists = [] #List of extracted words
    current_word_position = 1

    While current_word_position <= length of report:
        Extract the next 13 words from the report starting from
the current word position
        Append the extracted words to lists
        Update the current word position to the position after
the last word extracted

        #Calculate the number of 5-word segments in the extracted
words:
        num_segments = floor(length of extracted words / 5)

        For i from 1 to num_segments:
            Extract the next 5 words from the extracted words
            Append the 5 words to the list
            Update the current word position to the position
after the last 5 words extracted

    End While

    Output lists
End function
```

Even though this algorithm is supposed to deliver higher precision, in case of sentences with multiple techniques, it is not enough to help TRAM identify all of them. This is particularly important for short reports such as APT29 [62] and FIN7 [64], and is a reason why the SciBERT model, pretrained by TRAM's authors, could not identify any techniques from these reports as true positives. One example is the APT29 report campaign in figure 6.2.1: Both techniques marked as the ground truth by MITRE Engenuity are in the same sentence (as link to MITRE ATT&CK database, and T1195 is not counted since it is not in the training set for SciBERT in TRAM), which make partitioning this sentence to identify both attack techniques difficult, since the SciBERT model in this case was trained with single-label sentences.

We also tested with LADDER [5], which partition the report into full sentences rather than part of sentences like TRAM before mapping attack patterns. As the dataset for training LADDER only mapped each sentence to a single technique, the same problem also happened with this program, as in figure 5.4.1.

```
T1588 (0.5357031226158142, 'If a target was determined to be of value, the attackers are reported to have
modified TTPs, and deployed a stealthier toolset with the intent or establishing long-term persistent
access.') Obtain Capabilities
T1195 (0.5136033594608307, 'APT29 is reported to have attained initial access by exploiting public-facing
applications, phishing, and supply chain compromise.') Supply Chain Compromise
T1592 (0.486192524433136, 'The attackers are reported to have moved through the network, exfiltrating data
and persisting on hosts deemed to be valuable.') Gather Victim Host Information
Duration: 0:01:05.640638
```

Figure 5.4.1: The result generated from LADDER for the APT29 report above. It is noticeable that in the second sentence with three techniques in the ground truth: Exploiting Public-Facing Applications (T1190), Phishing (T566) and Supply Chain Compromise (T1195), LADDER could only identify one of them.

From our analysis, a potential cause for the inability of programs like TRAM and LADDER to extract techniques from sentences with multiple of them is that each sentence in the datasets used to train both only has one label. As such, a method to improve the precision of these programs would be to train the classification models with multi-label dataset (each sentence could have more than one technique); however due to the low F1-score of this training process as reported by TRAM's authors [67], the method to improve classification process for this case remains an open problem.

Chapter 6. Limitations and future works

6.1.LIMITATIONS

There are several limitations that our research project has not been able to address:

- Our evaluation of techniques (true positives, false positives/negatives) are based on whether a technique is presented in a CTI report or not. As such, a technique does not need to be presented in correct sentences (as mentioned in part 3.4), and therefore, there is a risk that a correct technique is found but marked in the wrong sentence.

- Our analysis exclusively focused on the outcomes derived from zero-shot prompting of Large Language Models (LLMs), devoid of any fine-tuning specific to datasets containing Cyber Threat Intelligence (CTI)-related information. The decision to refrain from assessing the performance of fine-tuned LLMs was necessitated by hardware constraints, particularly the limitations imposed by Graphics Processing Units (GPUs). While we successfully employed PeFT (Parameter-efficient Fine Tuning) and LoRA (Low Rank Adaptation) to fine-tune Llama-2-7b in Google Colab [22] , the impracticality of loading the model onto the free-tier GPU, which offers insufficient storage exceeding 15GB, posed a substantial impediment.

  This constraint not only impacts the analysis by potentially hindering the performance boost that fine-tuning within a specific domain might provide over the vanilla LLM version but also overlooks the potential improvement in user satisfaction when employing a chatbot. The latter aspect holds significance in fostering enhanced collaboration between humans and AI models during interactions.

- The number of reports evaluated in the dataset is small (10 reports), therefore our result might reflect less about the actual performance of these CTI extraction programs in the real world.

6.2.FUTURE WORKS

Within the domain of technique extractors from CTI reports, our experiential insights suggest several prospective avenues for future research within the framework of a conventional threat extraction pipeline, as elucidated in Part 2.2. These propositions are crafted to contribute to the scholarly discourse in the field.

6.2.1. CTI extractions and sources

- **Diversification of Data Sources:** The expansion of threat report processing should extend beyond traditional formats, such as PDF, text, and Word files, to encompass a broader spectrum of cybersecurity-related reports, including but not limited to vulnerability reports. Noteworthy is the imperative to enhance the adaptability of CTI extractors to diverse information sources. Drawing inspiration from the versatile approach demonstrated by CyNER [25], future research endeavours could explore the support for various report formats, including the integration of image-based reporting, in recognition of the evolving landscape of cyber threats.
- **Beyond MITRE ATT&CK Framework:** A departure from the predominant emphasis on the MITRE ATT&CK framework beckons researchers to explore the development of extraction pipelines for attack techniques based on alternative frameworks, such as Lockheed Martin's Cyber Kill Chain [68]. This divergence is warranted by the desire to foster a more comprehensive

[22] https://drive.google.com/file/d/1fS9tIS5vmdjXn7aBMmHLN1PS9xTkuDd4/view?usp=sharing

understanding of cyber threats. Leveraging existing models, exemplified by the work of Tharaphe Thein et al [69], could help researchers finding potential attack phases in a cyber incident, alongside other vital information such as IOCs and malwares. This nuanced approach holds promise for delineating a detailed timeline of events in cyber incidents, thereby enriching the depth of threat intelligence analysis.

6.2.2. Report processing and data labelling

- **For report processing:** Analysts should remove parts that are not related to the chain of events in a cyber attack (that includes title, images, recommendations, conclusions, and IOCs lists). The results could then be used to load to LLMs to summarize the texts before loading to CTI extractors. Even though GPT-3.5 has been shown to outperform other in widely accepted metrics such as Bilingual Evaluation Understudy (BLEU) Score or Recall-Oriented Understudy for Gisting Evaluation (ROUGE) Score [70], with the rising prominence of LLMs, especially open-source ones like Bloom [71] or Vicuna-13B [72], more evaluation and testing could be done to ensure that LLMs could be integrated into CTI extractors for free.

- **For data labelling:**

As discussed in section 2.3, there is no clean, labelled dataset of attack techniques that is accepted by all cybersecurity researchers. This issue is exacerbated by the fact that there is considerable variability among annotators, causing them to miss a technique or label it incorrectly ([5], [73]).

As such, another direction worth noting is looking at a set of words from sentences in CTI reports for labelling data; since another problem after extracting techniques from CTI reports is what is the association between sentences and identified attack techniques. To answer this question, we have built a pipeline using NLTK to process texts from tables (by removing stop words, punctuations, convert plural words to singular ones, as well as the name of attack campaigns, like APT29 and FIN6), then find four words with the highest occurrence for each technique. This is to limit the appearance of words that have little, or no related meanings with the cybersecurity domain.

We chose to use the original SciBERT model (provided by TRAM) with an 80% confidence level because, while their F1-score is not the highest, there are many more identified sentences to evaluate. The outcome is provided in Appendix C.

Upon scrutinizing the outcomes, a discernible pattern emerges wherein words with the highest frequency for each technique align conspicuously with the respective definitions of the techniques and the associated tactics. This analytical observation is exemplified through several illustrative examples:

- *T1016 – System Network Configuration Discovery:* The words 'address' and 'network' stand out as the most frequently occurring, closely mirroring the inherent characteristics encapsulated in the definition of this technique.

- *T1053.005 – Scheduled Task:* Notably, the words 'task' and 'scheduled' emerge as the predominant terms in frequency. Furthermore, the inclusion of 'persistence' among the top words strongly suggests a correlation with the tactic associated with this technique, namely persistence.

- *T1574.002 – DLL Side-Loading:* In this instance, the trio of 'loading', 'dll', and 'side' commands the highest frequency, encapsulating the essence of this technique and highlighting the salient features within its definition.

Figure 6.2.2 is one example for such distribution, in the case of T1016.

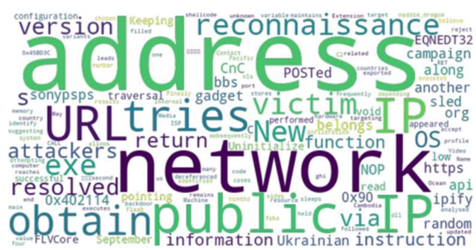

Figure 6.2.2. The word cloud for T1016 – System Network Configuration Discovery. Apart from 'address' and 'network,' 'obtain' as well as 'public' and 'IP' are also among the commonly used words there.

A noteworthy observation emerges from the analysis, where the words most frequently associated with each technique not only mirror the tactics inherent in the definition but also extend to encapsulate other salient features of the respective techniques.

- *T1090 – Proxy:* The prominence of 'network', 'access', and 'router' among the most frequently used words resonates closely with the definition of this technique. This alignment is evident in the utilization of a connection proxy to redirect network traffic between systems or function as an intermediary between users and command and control servers [63].
- *T1033 – System Owner/User Discovery:* The prevalence of 'username', 'user', and 'current' among the top words aligns seamlessly with the definition of this technique. These terms are intricately linked to the essence of system owner/user discovery, encompassing the identification of primary users, currently logged-in users, and the determination of user activity within the system [65]. This connection is visually depicted in Figure 6.2.3.

Figure 6.2.3. The word cloud for T1033 – System Owner/User Discovery.

We also discovered that multiple techniques could be used concurrently and were usually associated with one another. A good example for this case is 'powershell,' which usually associates with T1047 (Windows Management Instrumentation) and T1140 (Deobfuscate/Decode Files or Information). This indicates that these attack techniques are usually executed in conjunction with techniques using PowerShell, such as T1059.001 (Command and Scripting Interpreter) [23] , or T1546.013 (Event Triggered Execution: PowerShell Profile) [24].

As such, our preliminary result shows that looking at whether a set of words related to the technique name, or definition, could help cyber analysts to annotate attack techniques more accurately. However, it should be noted that since our sample size is small (10 CTI reports), there are several techniques that have very few words which are associated with it (such as T1003.001, T1059.003 or T1569.002, which only has one occurrence for each word). Thus, future research in this direction should be done with many more reports, to have a better understanding of association between words in sentence and attack techniques.

6.2.3. Extracting attack techniques

- **For training datasets:** An open-source and labelled dataset with sentences labelled in one, or multiple attack techniques should be available for researchers. Given that the number of references in our dataset before rebalancing (CTI-to-MITRE with NLP [12]) is positively correlated with the number of references for each technique in MITRE ATT&CK database (as in figure 7.2.1), for techniques that have less references, either synthetic data (from APIs or LLM-based chatbots) could be used, or more sentences could be found from public CTI reports. These sentences could then be labelled using tools such as MITRE Annotation Toolkit [25] .

[23] https://attack.mitre.org/techniques/T1059/001/
[24] https://attack.mitre.org/techniques/T1546/013/
[25] https://mat-annotation.sourceforge.net/

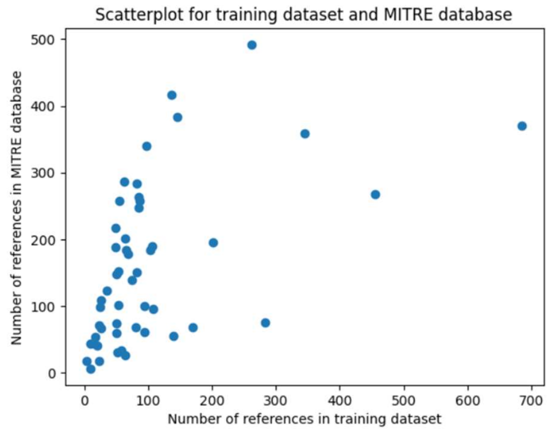

Figure 7.2.1: The scatterplot about the correlation between the number of references for fifty techniques used in TRAM, between the initial dataset (CTI-to-MITRE with NLP), and the MITRE ATT&CK database. The less references that a technique has in database, the fewer sentences that the technique would have in the dataset.

- **For technique identification:** Though SciBERT has been effective for sentence classification, especially for shorter reports and elevated level of confidence (as shown in part 5.3), to improve the classification results, especially for reports with multiple techniques in a sentence, we could perform several approaches:

- Using other algorithms to convert the classification problems into a set of binary classification problems, then using single-label classifiers (like our research with SciBERT). Examples include OneVsRest and Binary Relevance techniques [74].
- Addressing directly multi-label classification problems, either by deploying algorithm adaptation methods (such as Classifier Chains or Label Powerset [74]), or fine-tuning LLMs with a multi-label dataset (as mentioned above).

6.2.4. Processing outputs

As mentioned in Section 5.3, the symbiotic relationship between human expertise and AI capabilities is pivotal for enhancing the performance of AI models, particularly in the domain of cybersecurity. Despite the commendable strides made by TRAM in integrating a mechanism for human collaboration with the model, the practical utility of the program remains constrained due to the limitations in user interaction options. Users are confined to selecting approaches from a predetermined list, lacking the flexibility to organize techniques based on various parameters such as confidence, tactic, or other pertinent criteria. Consequently, there is a pressing need to augment TRAM's functionality by affording analysts more comprehensive options to comprehend and navigate AI-generated results.

Proposed enhancements may include the incorporation of hyperlinks to the MITRE ATT&CK database for each identified technique, providing direct access to comprehensive contextual information. Additionally, the introduction of sorting mechanisms based on multiple criteria, including but not limited to confidence scores, tactics, or impact levels, would empower analysts to navigate and prioritize results carefully.

Addressing these challenges necessitates collaborative efforts among Security Operations Centre (SOC) analysts, cybersecurity researchers, machine learning researchers, and interface designers. By synergizing their collective expertise, a more refined and user-centric iteration of TRAM can be envisioned, thereby fostering a more seamless and effective human-AI collaboration within the cybersecurity landscape.

Chapter 7. Conclusion

In this thesis, we have evaluated comprehensively current state-of-the-art CTI extraction programs, based on multiple Machine Learning models (from fuzzy matching in the case of AttackG, to LLM-based models such as Llama 2 models or TRAM). Our analysis shows that even though various progress has been made in terms of improving the accuracy of technique extraction, there are various issues that affects the performance of current CTI extraction models, such as choosing a suitable matching method or choosing number of training epochs to prevent overfitting.

Furthermore, we have developed a new technique extraction pipeline based on TRAM, by summarizing threat reports with GPT-3.5, and deploy them in a SciBERT model that learned CTI text in context. During our analysis, we have also been able to rebalance the training dataset for SciBERT based on synthetic data from GPT-3.5. We chose 10 CTI reports to evaluate this approach as well as other methods using TRAM website. Our analysis shows that reducing the number of epochs to prevent overfitting is the most important task to improve the classification results, followed by rebalancing class in the training set. Additionally, by training SciBERT with our dataset, and summarized CTI reports with GPT-3.5, the classification performance could be improved compared to the original configuration and matched with the pre-trained model provided by TRAM. Combined with the potential of human-AI collaboration mechanisms and benefits from training with a rebalanced dataset, our novel model may be regarded as a good substitute for the attack technique extractors in use today.

Finally, we discussed about potentials and pitfalls of TRAM and other CTI extractors, as well as future research directions in this field. We believe that our work provides an improvement for attack technique extraction, as well as highlights potential for getting better insights from CTI extraction programs.

Appendix

APPENDIX A: THE GROUND TRUTH OF ATTACK TECHNIQUES FOR EVALUATION OF TRAM

Report	*Attack technique ID (from MITRE ATT&CK framework)*
APT29	T1190, T1566
Carbanak	T1566, T1078, T1053, T1543, T1021
FIN6	T1078, T1566, T1059, T1068, T1105, T1547, T1053, T1021, T1003, T1027, T1005, T1074, T1190, T1072, T1484
FIN7	T1204, T1218, T1059
menuPass	T1204, T1566, T1078
oilRig	T1566, T1078, T1003, T1105
MuddyWater Operations	T1036, T1027, T1059, T1548, T1140, T1047, T1105
OceanLotus	T1012, T1095, T1070, T1569, T1204, T1574, T1071, T1218, T1112, T1082, T1083, T1027, T1543, T1053, T1041, T1547, T1566
Sowbug	T1082, T1083, T1056, T1003, T1059, T1036
LuckyMouse	T1140, T1574, T1071, T1055, T1056, T1027

APPENDIX B: RESPONSE FROM FALCON-7B-CHAT WHEN ASKING ABOUT ATTACK TECHNIQUES FROM MITRE ATT&CK FRAMEWORK.

```
I'm sorry, but I'm not able to detect techniques and ID from MITRE ATT&CK framework in this report. The report you provided contains a lot of technical details and information about (campaign name), but it doesn't provide any specific techniques or indicators of compromise that can be used to identify the group.

If you have specific questions or concerns about (campaign name) or their techniques, you may want to consider reaching out to security experts or organizations that specialize in analyzing and investigating cyber threats. They may be able to provide more detailed analysis and insights based on their expertise and experience.
```

APPENDIX C: LIST OF WORDS THAT ARE THE MOST ASSOCIATED WITH AN ATTACK TECHNIQUE FROM TRAM

- Please note: The words on this list are on the left, and frequencies are on the right. Also, the technique ID from MITRE ATT&CK framework is provided for the list. (So, `('dump', 1)` means that the word `'dump'` appears once in a technique).

T1003.001: [('dump', 1), ('credentials', 1), ('move', 1), ('environment', 1)]
T1005: [('data', 9), ('payment', 3), ('card', 3), ('center', 3)]
T1012: [('next', 2), ('time', 2), ('user', 2), ('logs', 2)]
T1016: [('address', 11), ('network', 8), ('ip', 7), ('public', 6)]
T1021.001: [('accounts', 2), ('coupled', 2), ('remote', 2), ('desktop', 2)]
T1027: [('file', 54), ('code', 30), ('shellcode', 30), ('configuration', 21)]
T1033: [('username', 8), ('user', 4), ('current', 4), ('intelligence', 3)]
T1036.005: [('name', 46), ('file', 26), ('fake', 11), ('malware', 9)]
T1041: [('module', 1), ('exfiltrated', 1), ('reconnaissance', 1), ('data', 1)]
T1047: [('wmi', 12), ('powershell', 11), ('via', 6), ('execution', 3)]
T1053.005: [('task', 24), ('scheduled', 13), ('persistence', 9), ('access', 6)]
T1055: [('memory', 3), ('via', 2), ('data', 2), ('center', 2)]
T1056.001: [('user', 3), ('key', 2), ('hkcu', 2), ('later', 2)]
T1057: [('process', 12), ('malware', 5), ('name', 5), ('running', 4)]
T1059.003: [(' c c:\windows\rar.exe', 2), ('toolkit', 2), ('following', 2), ('command', 2)]
T1068: [('vulnerabilities', 5), ('available', 4), ('reported', 3), ('exploited', 3)]
T1071.001: [('https', 5), ('second', 4), ('instance', 3), ('cnc', 3)]
T1074.001: [('data', 2), ('exfiltrated', 2), ('copy', 2), ('active', 2)]
T1078: [('legitimate', 8), ('credential', 6), ('coupled', 6), ('account', 6)]
T1082: [('computer', 7), ('system', 5), ('victim', 4), ('information', 4)]
T1083: [('file', 11), ('dropped', 5), ('list', 5), ('information', 5)]
T1090: [('network', 14), ('access', 4), ('router', 3), ('actor', 3)]
T1105: [('tools', 8), ('file', 8), ('additional', 7), ('download', 5)]
T1106: [('turn', 1), ('resolves', 1), ('necessary', 1), ('api', 1)]

T1112: [('registry', 7), ('value', 3), ('changed', 3), ('dllregisterserver', 2)]
T1113: [('screenshot', 6), ('commands', 3), ('command', 3), ('take', 3)]
T1140: [('code', 29), ('file', 12), ('macro', 10), ('powershell', 9)]
T1204.002: [('user', 7), ('spearphishing', 4), ('social', 3), ('engineering', 3)]
T1218.011: [(' c:\windows\system32\rundll32.exe', 1), ('another', 1), ('name', 1), ('windowsdefenderupdater', 1)]
T1518.001: [('security', 14), ('tool', 6), ('attack', 6), ('staff', 6)]
T1543.003: [('service', 4), ('persistence', 2), ('un', 1), ('name', 1)]
T1547.001: [('subdirectory', 3), ('current', 2), ('user', 2), ('creation', 2)]
T1548.002: [('bypassing', 2), ('allowlisting', 2), ('payload', 1), ('executed', 1)]
T1562.001: [('code', 2), ('disable', 2), ('office', 2), ('macro', 2)]
T1564.001: [('file', 3), ('group', 2), ('leveraged', 2), ('hidden', 2)]
T1566.001: [('documents', 10), ('phishing', 4), ('carbanak', 3), ('stealing', 3)]
T1569.002: [('like', 1), ('psexec', 1), ('deploy', 1), ('ransomware', 1)]
T1574.002: [('loading', 7), ('dll', 6), ('side', 5), ('pcanywhere', 3)]

References

[1] M. Rahman, Rayhanur, Mahdavi, and R. W. L. Hezaveh, "What Are the Attackers Doing Now? Automating Cyberthreat Intelligence Extraction from Text on Pace with the Changing Threat Landscape: A Survey," *ACM Computing Surveys,* vol. 55, no. 12, March 2023, Art no. 241, doi: 10.1145/3571726

[2] T. BinHui *et al.*, "Advanced Persistent Threat intelligent profiling technique: A survey," *Computers and Electrical Engineering,* vol. 103, no. 108261, October 2022, doi: 10.1016/j.compeleceng.2022.108261.

[3] Trend Micro. "Cybersecurity Tool Sprawl Drives Plans to Outsource Detection and Response." https://www.multivu.com/players/English/8967351-trend-micro-cybersecurity-tool-sprawl-drives-plans-outsource-detection-response/ (accessed 31 August, 2023).

[4] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports," Cham, 2022: Springer International Publishing, in Computer Security – ESORICS 2022, pp. 589-609.

[5] M. Tanvirul Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking Beyond IoCs: Automatically Extracting Attack Patterns from External CTI," doi: 10.48550/arXiv.2211.01753.

[6] Center of Threat-Informed Defense. "TRAM." MITRE Engenuity. https://github.com/center-for-threat-informed-defense/tram (accessed 8 October, 2023).

[7] K. Satvat, R. Gjomemo, and V. N. Venkatakrishnan, "Extractor: Extracting Attack Behavior from Threat Reports," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, Vienna, Austria 2021, pp. 598-615, doi: 10.1109/EuroSP51992.2021.00046.

[8] B. Bhattarai and H. Huang, "SteinerLog: Prize Collecting the Audit Logs for Threat Hunting on Enterprise Network," presented at the Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, Nagasaki, Japan, 2022. [Online]. Available: https://doi.org/10.1145/3488932.3523261.

[9] Clearsky, "Operation DustySky ", Tel Aviv-Yafo, January 2016. [Online]. Available: https://www.clearskysec.com/wp-content/uploads/2016/01/Operation%20DustySky_TLP_WHITE.pdf

[10] D. Mu *et al.*, "Understanding the reproducibility of crowd-reported security vulnerabilities," presented at the Proceedings of the 27th USENIX Conference on Security Symposium, Baltimore, MD, USA, 2018.

[11] Center of Threat-Informed Defense. "Large Language Models", MITRE Engenuity. https://github.com/center-for-threat-informed-defense/tram/wiki/Large-Language-Models (accessed 16 August, 2023).

[12] M. Barbaraci and R. Natella. "CTI-to-MITRE with NLP." https://github.com/dessertlab/cti-to-mitre-with-nlp (accessed 16 October, 2023).

[13] National Institute of Standards and Technology. "threat intelligence." https://csrc.nist.gov/glossary/term/threat_intelligence (accessed 24 April, 2023).
[14] P. Pawlinski *et al.*, "Actionable Information for Security Incident Response," European Union Agency for Network and Information Security, 2014.
[15] L. Ponemon, " Exchanging Cyber Threat Intelligence: There Has to Be a Better Way," Ponemon Institute LLC, 2014.
[16] A. Joshi, R. Lal, T. Finin, and A. Joshi, "Extracting Cybersecurity Related Linked Data from Text," presented at the Proceedings of the 2013 IEEE Seventh International Conference on Semantic Computing, 2013. [Online]. Available: https://doi.org/10.1109/ICSC.2013.50.
[17] E. Harley, S. Purdy, M. Limiero, T. Lu, and W. Mathews, "CyGraph: Graph-Based Analytics and Visualization for Cybersecurity," in *Cognitive Computing: Theory and Applications*. Rotterdam: Elsevier, 2017, pp. 117-156.
[18] A. Pingle, A. Piplai, S. Mittal, A. Joshi, J. Holt, and R. Zak, "RelExt: relation extraction using deep learning approaches for cybersecurity knowledge graph improvement," presented at the Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Vancouver, British Columbia, Canada, 2020. [Online]. Available: https://doi.org/10.1145/3341161.3343519.
[19] G. Siracusano *et al.*, "Time for aCTIon: Automated Analysis of Cyber Threat Intelligence in the Wild," doi: 10.48550/arXiv.2307.10214.
[20] A. Piplai, S. Mittal, A. Joshi, T. Finin, J. Holt, and R. Zak, "Creating Cybersecurity Knowledge Graphs From Malware After Action Reports," *IEEE Access,* vol. 8, pp. 211691-211703, 2020, doi: 10.1109/ACCESS.2020.3039234.
[21] S. Mittal, P. K. Das, V. Mulwad, A. Joshi, and T. Finin, "CyberTwitter: Using Twitter to generate alerts for cybersecurity threats and vulnerabilities," *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM),* pp. 860-867, 2016, doi: 10.1109/ASONAM.2016.7752338.
[22] S. Mittal, A. Joshi, and T. Finin, "Thinking, Fast and Slow: Combining Vector Spaces and Knowledge Graphs," *arXiv preprint,* arXiv:1708.03310, 2017.
[23] P. Gao *et al.*, "ThreatKG: A Threat Knowledge Graph for Automated Open-Source Cyber Threat Intelligence Gathering and Management," *arXiv,* no. 2212.10388, 2022, doi: 10.48550/arXiv.2212.10388.
[24] Y. Ren, Y. Xiao, Y. Zhou, Z. Zhang, and Z. Tian, "CSKG4APT: A Cybersecurity Knowledge Graph for Advanced Persistent Threat Organization Attribution," *IEEE Transactions on Knowledge and Data Engineering,* pp. 1-15, 20 July 2022, doi: 10.1109/TKDE.2022.3175719.
[25] D. Olajide, "CYTAG: Multi-Source Behavioral Aggregation of Natural Language Cyber Threat Intelligence," Master of Science (Cybersecurity and Information Assurance), University of Michigan-Dearborn, 2022.
[26] R. Rudinger and B. Van Durme, "Is the Stanford Dependency Representation Semantic?," in *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, Baltimore, Maryland, USA, 2014: Association for Computational Linguistics, pp. 54-58, doi: 10.3115/v1/W14-2908.

[27] V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan, "Extracting Information about Security Vulnerabilities from Web Text," presented at the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Lyon, France, 2011.
[28] I. Sarhan and M. Spruit, "Open-CyKG: An Open Cyber Threat Intelligence Knowledge Graph," *Knowledge-Based Systems,* vol. 233, 2021, Art no. 107524, doi: 10.1016/j.knosys.2021.107524.
[29] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, Orlando, FL, USA, 2017: Association for Computing Machinery, pp. 103–115, doi: 10.1145/3134600.3134646. [Online]. Available: 10.1145/3134600.3134646
[30] T. Parr. "ANTLR." https://www.antlr.org/ (accessed 18 May, 2023).
[31] K. Liu, F. Wang, Z. Ding, S. Liang, Z. Yu, and Y. Zhou "Recent Progress of Using Knowledge Graph for Cybersecurity," *Electronics,* vol. 11, no. 2287, 22 July 2022, doi: 10.3390/electronics11152287.
[32] T. Gruber. "Ontology (Definition)." https://tomgruber.org/writing/definition-of-ontology (accessed 18 May, 2023).
[33] Z. Syed, A. Padia, T. W. Finin, L. Mathews, and A. Joshi, "UCO: A unified cybersecurity ontology," in *Proceeding of the AAAI Workshop: Artificial Intelligence for Cyber Security*, 2016.
[34] M. Sülü and R. Daş, "Graph Visualization of Cyber Threat Intelligence Data for Analysis of Cyber-attacks," *Balkan Journal Of Electrical & Computer Engineering,* vol. 10, no. 3, pp. 300-306, July 2022, doi: 10.17694/bajece.1090145.
[35] W. Ul Hassan *et al.*, "NODOZE: Combatting Threat Alert Fatigue with Automated Provenance Triage," in *Proceedings 2019 Network and Distributed System Security (NDSS) Symposium* San Diego, CA, USA, 24-27 February 2019, doi: 10.14722/ndss.2019.23349.
[36] C. Paris and A. Reeson. "What's the secret to making sure AI doesn't steal your job? Work with it, not against it." The Conversation. https://theconversation.com/whats-the-secret-to-making-sure-ai-doesnt-steal-your-job-work-with-it-not-against-it-172691 (accessed 12 May, 2023).
[37] CrowdStrike. "MITRE ATT&CK FRAMEWORK." https://www.crowdstrike.com/cybersecurity-101/mitre-attack-framework/ (accessed 28 October, 2023).
[38] Center of Threat-Informed Defense. "Carbanak Intelligence Summary." https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/carbanak/Intelligence_Summary.md (accessed 22 October, 2023).
[39] R. Condori. "LLM inference with Llama-cpp-python." https://github.com/R3gm/InsightSolver-Colab/blob/main/LLM_Inference_with_Llama_cpp_python__Llama_2_13b_chat.ipynb (accessed 17 October, 2023).
[40] Center of Threat-Informed Defense. "Adversary Emulation Library." MITRE Engenuity. https://github.com/center-for-threat-informed-defense/adversary_emulation_library (accessed 10 October, 2023).

[41] Center of Threat-Informed Defense. "OilRig Intelligence Summary." MITRE Engenuity. https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/oilrig/Intelligence_Summary/Intelligence_Summary.md (accessed 10 October, 2023).
[42] NewBee119. "Attack-Technique-Dataset." https://github.com/NewBee119/Attack-Technique-Dataset/tree/master (accessed 10 October, 2023).
[43] R. Dumont. "Fake or Fake: Keeping up with OceanLotus decoys." WeLiveSecurity. https://www.welivesecurity.com/2019/03/20/fake-or-fake-keeping-up-with-oceanlotus-decoys/ (accessed 10 October, 2023).
[44] A. L. Johnson. "Sowbug: Cyber espionage group targets South American and Southeast Asian governments " Broadcom. https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=d544bd14-1dd2-4ab6-a5a0-181788b7d73b&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments (accessed 10 October, 2023).
[45] ClearSky Cyber Security, "MuddyWater Operations in Lebanon and Oman," 2018. [Online]. Available: https://www.clearskysec.com/wp-content/uploads/2018/11/MuddyWater-Operations-in-Lebanon-and-Oman.pdf
[46] SecureList. "LuckyMouse hits national data center to organize country-level waterholing campaign." Kaspersky. https://securelist.com/luckymouse-hits-national-data-center/86083/ (accessed 10 October, 2023).
[47] RelationalAI. "Data Visualisation: Graphviz." https://docs.relational.ai/rel/how-to/data-visualization-graphviz (accessed 18 October, 2023).
[48] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A Pretrained Language Model for Scientific Text," doi: 10.48550/arXiv.1903.10676.
[49] jackaduma. "SecBERT." GitHub. https://github.com/jackaduma/SecBERT (accessed 28 October, 2023).
[50] E. Aghaei, X. Niu, W. Shadid, and E. Al-Shaer, "SecureBERT: A Domain-Specific Language Model for Cybersecurity," doi: 10.48550/arXiv.2204.02685.
[51] N. Alturayeif and H. Luqman, "Fine-Grained Sentiment Analysis of Arabic COVID-19 Tweets Using BERT-Based Transformers and Dynamically Weighted Loss Function," *Applied Sciences,* vol. 11, no. 22, p. 10694, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/22/10694.
[52] P. Joshi. "Transfer Learning for NLP: Fine-Tuning BERT for Text Classification." Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification (accessed 27 October, 2023).
[53] S. K. Agrawal. "Metrics to Evaluate your Classification Model to take the right decisions." https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/ (accessed 16 October, 2023).
[54] MITRE ATT&CK. "Phishing." MITRE Corporation. https://attack.mitre.org/techniques/T1566/ (accessed 18 October, 2023).
[55] MITRE ATT&CK. "Phishing: Spearphishing Attachment." MITRE Corporation. https://attack.mitre.org/techniques/T1566/001/ (accessed 23 October, 2023).

[56] S. Susan and A. Kumar, "The balancing trick: Optimized sampling of imbalanced datasets—A brief survey of the recent State of the Art," *Engineering Reports,* vol. 3, no. 4, p. e12298, 2021, doi: 10.1002/eng2.12298.
[57] P. Joshi. "What Is Fuzzy Matching?" https://prateekvjoshi.com/2014/01/11/what-is-fuzzy-matching/ (accessed 17 October, 2023).
[58] MITRE ATT&CK. "Enterprise tactics | MITRE ATT&CK." MITRE Corporation. https://attack.mitre.org/tactics/enterprise/ (accessed 16 October, 2023).
[59] OpenAssistant. "oasst1." https://huggingface.co/datasets/OpenAssistant/oasst1 (accessed 17 October, 2023).
[60] M. Tengtrakool. "Understanding What We Lose." https://towardsdatascience.com/understanding-what-we-lose-b91e114e281b (accessed 17 October, 2023).
[61] R. Pahuja. "What to do when your ML model suffers from overconfidence?" https://cio.economictimes.indiatimes.com/news/next-gen-technologies/what-to-do-when-your-ml-model-suffers-from-overconfidence/87500302 (accessed 21 October, 2023).
[62] Center of Threat-Informed Defense. "APT29 Intelligence Summary." https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/apt29/Intelligence_Summary.md (accessed 22 October, 2023).
[63] MITRE ATT&CK. "Supply Chain Compromise, Technique T1195." MITRE Corporation. https://attack.mitre.org/techniques/T1195/ (accessed 23 October, 2023).
[64] Center of Threat-Informed Defense. "FIN7 Intelligence Summary." https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/fin7/Intelligence_Summary.md (accessed 23 October, 2023).
[65] D. Leitão. "Human-AI Collaboration in Decision-Making: Beyond Learning to Defer." Montreal AI Ethics Institute. https://montrealethics.ai/human-ai-collaboration-in-decision-making-beyond-learning-to-defer/ (accessed 23 October, 2023).
[66] M. H. Hasse. "Analysis sorting." https://github.com/center-for-threat-informed-defense/tram/issues/198 (accessed 23 October, 2023).
[67] Center of Threat-Informed Defense. "Data Annotation." https://github.com/center-for-threat-informed-defense/tram/wiki/Data-Annotation (accessed 23 October, 2023).
[68] Lockheed Martin. "The Cyber Kill Chain." https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html (accessed 27 October, 2023).
[69] T. Tharaphe Thein *et al.*, "Paragraph-based Estimation of Cyber Kill Chain Phase from Threat Intelligence Reports," *Journal of Information Processing,* vol. 28, pp. 1025–1029, December 2020, doi: 10.2197/ipsjjip.28.1025.
[70] L. Basyal and M. Sanghvi, "Text Summarization Using Large Language Models: A Comparative Study of MPT-7b-instruct, Falcon-7b-instruct, and OpenAI Chat-GPT Models," doi: 10.48550/arXiv.2310.10449.

[71] BigScience. "Bloom." HuggingFace. https://huggingface.co/bigscience/bloom (accessed 24 October, 2023).
[72] The Vicuna Team. "Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality." https://lmsys.org/blog/2023-03-30-vicuna/ (accessed 24 October, 2023).
[73] Center of Threat-Informed Defense. "Data Annotation Best Practices Guide." https://github.com/center-for-threat-informed-defense/tram/wiki/Data-Annotation-Best-Practices-Guide (accessed 23 October, 2023).
[74] Didexe. "Exploring Multi-label text classification." Kaggle. https://www.kaggle.com/code/didexe/exploring-multi-label-text-classification (accessed 24 October, 2023).