DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Towards Cybersecurity Dataset Creation An LLM-Assisted Approach for Named Entity Recognition in Threat Intelligence

Yuri do Amaral Nobre Maia

Programa de Pós-Graduação Profissional em Engenharia Elétrica
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Towards Cybersecurity Dataset Creation An LLM-Assisted Approach for Named Entity Recognition in Threat Intelligence

Criação de Conjuntos de Dados de Cibersegurança Uma Abordagem Assistida por LLM para Reconhecimento de Entidades Nomeadas em Inteligência de Ameaças

Yuri do Amaral Nobre Maia

Orientador: Demétrio Antônio da Silva Filho, Dr.
Coorientador: Robson de Oliveira Albuquerque, Dr.

DISSERTAÇÃO DE MESTRADO PROFISSIONAL EM
ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPEE.MP.095
BRASÍLIA/DF: AGOSTO – 2025

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Towards Cybersecurity Dataset Creation An LLM-Assisted Approach for Named Entity Recognition in Threat Intelligence

Yuri do Amaral Nobre Maia

Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Demétrio A. da Silva Filho, Ph.D, FT/UnB _____
Orientador

Prof. João José Costa Gondim, Ph.D, FT/UnB _____
Examinador interno

Prof. Cristoffer Leite da Silva , Ph.D, TU/e _____
Examinador externo

Prof. Fábio Lúcio Lopes de Mendonça, Ph.D, FT/UnB _____
Suplente

Prof. Robson de Oliveira Albuquerque, Ph.D, FT/UnB _____
Convidado de Honra

FICHA CATALOGRÁFICA

MAIA, YURI DO AMARAL NOBRE
Towards Cybersecurity Dataset Creation An LLM-Assisted Approach for Named Entity Recognition in Threat Intelligence [Distrito Federal] 2025.
xvi, 60 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2025).
Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. BERT	2. Gemini
3. Named Entity Recognition	4. STIX
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

MAIA, Y.A.N. (2025). *Towards Cybersecurity Dataset Creation An LLM-Assisted Approach for Named Entity Recognition in Threat Intelligence*. Dissertação de Mestrado Profissional, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 60 p.

Yuri do Amaral Nobre Maia
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

ACKNOWLEDGEMENTS

I would like to acknowledge my supervisors, Dr. Demetrio Antônio da Silva Filho and Dr. Robson de Oliveira Albuquerque, who inspired and supported me throughout the research. Dr. Rafael Rabelo Nunes, whose light-hearted conversations stirred my outlook.

I would like to thanks my family for supporting me, as well as my friends and fellow students who lengthily discussed my research even in their rest time.

ABSTRACT

The increasing number of cybersecurity reports poses a challenge to efficiently retrieving and sharing Cyber Threat Intelligence. However, publicly available cybersecurity datasets for Natural Language Processing (NLP) remain scarce, hindering progress in automated intelligence production. To tackle this challenge, this article presents Yet Another Cybersecurity Database (YACSDB), a dataset designed to enhance Named Entity Recognition (NER) using Structured Threat Information Expression (STIX) entities for interoperability. Our pipeline extracts STIX Domain Objects from unstructured reports, leveraging Google's Gemini and Bidirectional Encoder Representations from Transformers (BERT) model to assist in labeling and reduce resource needs. The dataset uses Inside–Outside–Beginning (IOB) notation to facilitate fine-tuning in sequence tagging tasks. Reports were selected for representativeness across different years. To the best of our knowledge, it is among the largest cybersecurity NER datasets with temporal information annotated by a single machine-assisted annotator. To evaluate the dataset, we fine-tuned seven BERT models to demonstrate its effectiveness for NER. The results emphasize the importance of domain-specific datasets in cybersecurity NLP and highlight key challenges. YACSDB serves as a benchmark for model comparison, solution development, and knowledge graph generation. It is publicly available to foster future research in cybersecurity NLP.

RESUMO

O número crescente de relatórios de cibersegurança representa um desafio para a recuperação e o compartilhamento eficientes de Inteligência de Ameaças Cibernéticas (CTI). No entanto, conjuntos de dados de cibersegurança publicamente disponíveis para Processamento de Linguagem Natural (PLN) permanecem escassos, dificultando o avanço na produção automatizada de inteligência. Para enfrentar este desafio, este artigo apresenta o Yet Another Cybersecurity Database (YACSDB), um conjunto de dados projetado para aprimorar o Reconhecimento de Entidades Nomeadas (REN) utilizando entidades Structured Threat Information Expression (STIX) para interoperabilidade. Nosso pipeline extrai Objetos de Domínio STIX de relatórios não estruturados, aproveitando os modelos Gemini do Google e Bidirectional Encoder Representations from Transformers (BERT) para auxiliar na rotulagem e reduzir a necessidade de recursos. O conjunto de dados emprega a notação Inside–Outside–Beginning (IOB) para facilitar o fine-tuning em tarefas de etiquetagem de sequências. Os relatórios foram selecionados visando a representatividade ao longo de diferentes anos. Pelo nosso conhecimento, este é um dos maiores conjuntos de dados de REN em cibersegurança com informações temporais, anotado por um único anotador assistido por máquina. Para avaliar o conjunto de dados, realizamos o fine-tuning de sete modelos BERT para demonstrar sua eficácia para REN. Os resultados enfatizam a importância de conjuntos de dados específicos de domínio em PLN de cibersegurança e destacam desafios importantes. O YACSDB serve como um benchmark para comparação de modelos, desenvolvimento de soluções e geração de grafos de conhecimento. Ele está publicamente disponível para fomentar pesquisas futuras em PLN de cibersegurança.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUCTION

The current cybersecurity landscape is characterized by an escalating threat environment, exemplified by sophisticated ransomware attacks targeting critical infrastructure and large-scale data breaches affecting thousands (9). The expansion of digital systems and interconnected technologies increases the attack surface, creating many opportunities for malicious actors to exploit vulnerabilities. This growth can be noted from the year-over-year increase in vulnerability discoveries (10). This is the scenario that acquired the attention of multiple stakeholders, including academic researchers, cybersecurity practitioners, and threat actors themselves. As a consequence, many organizations regularly publish cybersecurity reports that provide analysis of emerging threats, attack vectors, and defense strategies.

Cybersecurity is an expanding domain in which the high incidence of unstructured data presents challenges for analysis. Consequently, treating this data demands large volumes of diverse data (11). Particularly, advances in Natural Language Processing (NLP) enhanced the automation of information extraction from cybersecurity reports (12), which is exploited in this specialized domain.

Traditional NLP techniques have been employed for different goals in the cybersecurity domain. This scenario changed with the creation of groundbreaking architectures, such as Transformers (5), which enticed the development of state-of-the-art Large Language Models (LLMs) that have reshaped numerous NLP tasks. These models enable machines to extract deep, context-aware information with high accuracy. For cybersecurity analysts, these advances facilitate the automation of data processing (12) and enable analysis using standardized languages (13).

Within specialized domains, Bidirectional Encoder Representations from Transformers (BERT) (7) has demonstrated excellent performance even with limited resources. The model's capacity for fine-tuning across diverse NLP tasks leverages transfer learning, requiring smaller datasets for domain-specific applications. This approach has been explored in specialized fields, as with SciBERT (14) for scientific texts, BioBERT (15) for biomedical text processing, and FinBERT (16) for financial domain applications. Their results show potential for cybersecurity applications, yet the development of domain-specific BERT for the cybersecurity domain is limited.

There are two straightforward strategies for adapting an LLM model for a domain. The first approach involves pre-training the model on a domain-specific corpus. In this approach of Domain Adaptation Pre-Training (DAPT) (16), models learn specialized linguistic patterns and domain knowledge that may differ from general-purpose datasets. This is particularly important in the cybersecurity domain, where words may have different semantics than usual. However, DAPT requires many computational resources and large corpora, which may be difficult to arrange. Within the cybersecurity domain, some researchers delved into this and implemented different models, such as CyBERT (12) and SecureBERT (17). The second strategy involves fine-tuning for a specific task using labeled datasets within the targeted domain. This generally employs supervised learning and, although it needs a great amount of data, it still requires substantially less than what is needed for DAPT. While it is true that models may not fully benefit from their pre-training in specific domains, this is not a concern for cybersecurity, as there are DAPT models available.

As technological infrastructure continues to evolve, new information sources emerge. Social media platforms generate a large amount of cybersecurity content; therefore, Open-Source Intelligence (OSINT) grows in importance and plays a key role in Cyber Threat Intelligence (CTI). One key objective of CTI frameworks is the sharing of information regarding existing or evolving cyber attacks, thereby allowing preventive measures (11). Nonetheless, much of the shared data comes in different formats and contains unstructured data (18), posing a challenge for consumption. Named Entity Recognition (NER) techniques play an important role in this data.

Training an LLM for NER requires large labeled datasets to learn subtle patterns in languages. Although many large textual corpora are employed in LLM pre-training, most of them contain text whose meaning may differ in the cybersecurity domain (19). Anyhow, the volume of cybersecurity text available is high. However, despite the extensive amount of cybersecurity reports, publicly available NLP datasets in the cybersecurity domain remain scarce, as it is noted in studies (20, 21, 12, 9, 22, 23).

This research aims to address this data gap. We present a methodology for the generation of labeled datasets and introduce YACSDB$_{NER}$, a public dataset for Named Entity Recognition in the cybersecurity domain. The dataset is structured around the STIX language to provide CTI in a standardized language to allow efficient sharing of information.

1.1 OBJECTIVES

This research's main goal is to address the scarcity of publicly labeled datasets in cybersecurity NLP applications to enable the employment of supervised learning strategies and fair comparison of existing models. Taking into consideration the identified gaps in the NER task applied to this domain, the research's main question is: *How to design and efficiently build a dataset for NER in the cybersecurity domain*?

To better understand the problem, this question was broken down into three other research questions:

1. **RQ1.** *What are the main goals to which the datasets are designed in the cybersecurity domain?*
2. **RQ2.** *Which are the common constraints in the dataset generation?*
3. **RQ3.** *How to efficiently build the dataset?*

There are many NER objectives in cybersecurity; therefore, we identified and analyzed different goals in existing research. Once the goals were identified, we selected key named entities related to STIX Domain Objects, as they represent key concepts that a cybersecurity analyst may encounter when describing a threat. After it, aspects of the corpus representativeness, as well as its format to align with its application to BERT models, were taken into consideration. Following it, we developed a semi-automated labeling process to leverage different LLMs to generate high-quality annotated datasets.

1.2 RESEARCH CONTRIBUTIONS

This study advances the landscape of datasets for NLP in cybersecurity and contributes to broadening research in this domain. Our main contributions are outlined as follows:

- Development of a novel labeling pipeline that integrates Bidirectional Encoder Representations from Transformers (BERT) and Gemini models to preprocess, analyze, and compile cybersecurity reports into a comprehensive annotated dataset;
- Introduction of Yet Another Cybersecurity Database for Named Entity Recognition (YACSDB$_{NER}$), a publicly available dataset comprising annotated text spans in IOB notation designed for NER of STIX entities;
- Comprehensive evaluation of YACSDB$_{NER}$ performance accross the main domain-adapted BERT-based models in cybersecurity.
- Comparative analysis of seven BERT-based approaches for NLP in the cybersecurity domain, providing insights into the models' performance and establishing benchmarks for future research in the domain.

YACSDB$_{NER}$ is available on GitHub <https://github.com/boutdatansec/YACSDB>.

1.3 OUTLINE

The remainder of the paper is organized as follows. First, Chapter 2 sets the theoretical framework and explores related work on the NER task in cybersecurity and available datasets. Building on this foundation, Chapter 3 describes the pipeline to achieve the final dataset. Chapter 4 details the setup for evaluating the dataset and presents the findings and discussion over the results. Finally, the conclusion of this study is included in Chapter 5 along with future works.

2 BACKGROUND AND RELATED WORKS

The dynamic, complex nature of the cybersecurity landscape requires the development of robust methods to automate the acquisition of threat intelligence. This chapter introduces the groundwork for understanding the methods of Natural Language Processing and Deep Learning applied to the cybersecurity domain.

We initiate by detailing the intelligence cycle, which is the backbone of this study. The section details the main sources of intelligence to better comprehend what Cyber Threat Intelligence (CTI) is. Following this, we explore prevalent CTI standards and frameworks, including the Cyber Kill Chain, Diamond Model, and MITRE ATT&CK, MAEC, CAPEC, and STIX, which provide a structured understanding of cyber threats and the entities that define them. Subsequently, we investigate the evolution of techniques in Natural Language Processing (NLP), delving into Deep Neural Networks' capabilities, and clarifying how these techniques enable machines to comprehend and process human language. The following section presents a review of Named Entity Recognition in cybersecurity, examining the methodologies employed in existing NER models in the cybersecurity domain. Finally, the chapter concludes with a comprehensive review of available datasets for NER in cybersecurity, assessing their characteristics, limitations, and suitability for training and evaluating robust NER systems. Through this detailed exploration, this chapter aims to establish a solid theoretical and practical foundation for the subsequent research and development in named entity recognition for cyber threat intelligence. We initiate by detailing the intelligence cycle, which is the backbone of this study. The section details the main sources of intelligence to better comprehend what Cyber Threat Intelligence (CTI) is. Following this, we explore prevalent CTI standards and frameworks, including the Cyber Kill Chain, Diamond Model, and MITRE ATT&CK, MAEC, CAPEC, and STIX, which provide a structured understanding of cyber threats and the entities that define them. Subsequently, we investigate the evolution of techniques in Natural Language Processing (NLP), delving into Deep Neural Networks' capabilities, and clarifying how these techniques enable machines to comprehend and process human language. The following section presents a review of Named Entity Recognition in cybersecurity, examining the methodologies employed in existing NER models in the cybersecurity domain. Finally, the chapter concludes with a comprehensive review of available datasets for NER in cybersecurity, assessing their characteristics, limitations, and suitability for training and evaluating robust NER systems. Through this detailed exploration, this chapter aims to establish a solid theoretical and practical foundation for the subsequent research and development in named entity recognition for cyber threat intelligence.

2.1 INTELLIGENCE CYCLE

Intelligence is actionable information intended to guide decisions and change outcomes. The Brazilian Intelligence Agency defines it as the production of knowledge in order to support more informed decision making by its users (24). To achieve its objective, actions of data collection and analysis are performed, which allow its comprehension. The information gathered in this process must be actionable by the user

Figure 2.1: The Intelligence Process (1).

to make decisions (2). A similar definition of Intelligence can be understood as the product of analyzed information about the operational environment to anticipate future situations and inform decisions (1).

The objectives of Intelligence must be aligned with the user's needs to be valuable. The Intelligence cycle is the process to ensure that the results meet the requirements. It consists of six interrelated steps (1) that are shortly described as follows:

1. **Planning and direction**: development of plans and goals to support the mission;
2. **Collection**: data acquisition and related activities to address the intelligence requirements;
3. **Processing and exploitation**: conversion of raw collected data into a usable format to be consumed by the user;
4. **Analysis and production**: intelligence product from the compilation, correlation, and refinement of the gathered information and other intelligence received;
5. **Dissemination and integration**: delivery of the intelligence to the end user on the necessary means;
6. **Evaluation and feedback**: continuous assessment of the intelligence process to adjust any step.

The Intelligence Cycle is a customer-oriented process. Every stage must aim to satisfy the end user requirements (10). The execution is not a strict waterfall process, where one step only starts when the previous has ended, but rather an integrated and collaborative process with timely feedback amongst stages, as seen in Figure 2.1. This adaptive structure allows it to follow the customer priorities and requirements and even reevaluate them if needed.

There are three different levels of intelligence. This distinction is made to attend to different decision process levels. Each level has different goals, sources, customers, and formats. Typically, the levels are divided into strategic, tactical, and operational.

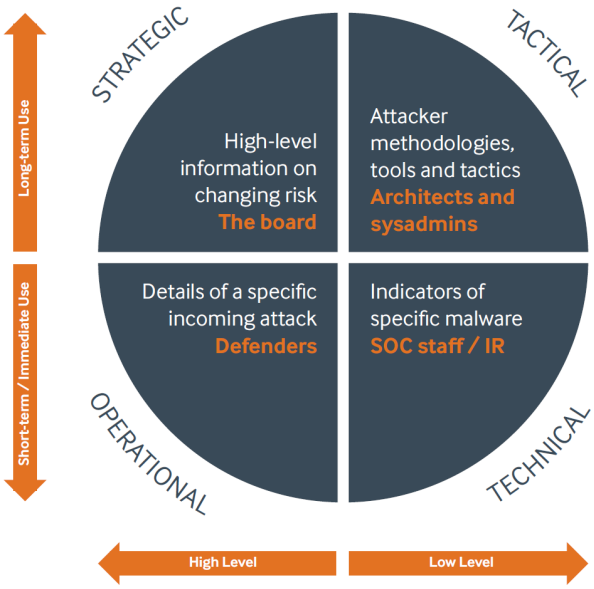


Figure 2.2: Levels of Threat Intelligence (2).

The levels are hierarchical to provide different levels of information. The hierarchy order is different depending on the context, which will be split into two for the sake of clarification. In military doctrine, the intelligence levels are related to warfare levels and are ordered, from higher to lower, strategic, operational, and tactical (1). Strategic Intelligence is directly related to the national objectives and long-term interests. Operational Intelligence provides intelligence related to major operational objectives and analysis of the operational environment. Tactical Intelligence supports the planning and execution of engagements and missions.

From a business perspective, the levels can be ordered strategic, tactical, and operational, with a possible technical level. The categorization depends on temporal relevance and abstraction level, which can be characterized as in Figure 2.2. In this definition, Strategic Intelligence is consumed by high-level strategists within an organization to guide business decisions and help to understand the impact of their choices. Tactical Intelligence supports managers and heads of areas to execute the strategy and achieve the objectives in a medium-term time horizon. Operational Intelligence focuses on specific issues of process execution of daily operations. Technical Intelligence is related to the direct treatment of raw data to rapidly share intelligence of immediate impact in daily activities, which can be part of the Operational Intelligence cycle.

One approach to studying Intelligence is by examining its disciplines, or intelligence sources. The list of basic disciplines may vary among some authors. Traditionally, we can define the following: Open-Source Intelligence (OSINT), Human Intelligence (HUMINT), Signals Intelligence (SIGINT), Geospatial Intelligence (GEOINT), Measurement and Signature Intelligence (MASINT), and Technical Intelligence (TECHINT) (1, 25). These disciplines have parallel activities with different emphasis for each.

Among the fundamental intelligence disciplines, OSINT is one of the most accessible and readily available intelligence disciplines, because its collection and analysis do not require advanced technology and extensive resources (25). Given these characteristics, we will further explore its understanding.

2.1.1 Open-Source Intelligence

The usage of the acronym OSINT can be traced to 1990, though authors have noted that it has been present for centuries. To differentiate the mere observation of facts in the world from OSINT, we will adopt the definition that aligns it with other intelligence disciplines and the intelligence cycle. Open-Source Intelligence is "the methodical collection and exploitation of information from publicly available sources to fulfill an intelligence requirement" (26).

Historically, the rise of OSINT is most clear through these lenses with the expansion of public newspapers and radio broadcasts (26). With the development of the Internet and computer networks, cyberspace has become an important information environment, which is part of the acknowledged cyber domain (27).

Many activities that already existed in the physical world take advantage of the existing technology to create a new cyber-enabled version. This has driven the surge of new technologies and environments that are sources of information. The rise of social media importance in the cyber domain greatly modified the scenario of OSINT, which has been pointed as a second generation of OSINT (28). Besides the cyber-enabled activities, cyber capabilities evolved to a point of entire operations being executed in cyberspace (27), thus justifying what can be called Cyber Intelligence. Large amounts of data are generated daily on platforms such as X (formerly Twitter), Facebook, and Instagram, as well as on companies' newsletters, individual blogs, and public articles. Collection and analysis are both accessible and complex in this scenario. To adapt to it, new methodologies and technologies have been developed, as well as the aggregation of areas like big data, data mining, software development, and data science.

OSINT in cyberspace is widely leveraged by governments, law enforcement, and intelligence services to investigate and fight against cybercrime and organized crime (29). Nevertheless, private organizations also take great advantage of it, as many use digital systems and communication. Understanding threats to defend systems from cyberattacks is the main goal of Cyber Threat Intelligence (CTI).

2.1.2 Cyber Threat Intelligence

The intelligence cycle conducted to counter threats in cyberspace against an organization is mentioned as Cyber Threat Intelligence. It is a subset of the Cyber Intelligence that assesses possible threats to counter or respond to them (30). It is an intelligence focused on the adversary to provide intelligence products concerning hostile cyber organizations and cyber forces capabilities (31). The primary objective is, thus, acquiring knowledge about adversaries and their motivations, intentions, and methods to protect critical assets (32).

Most of the cyber activity is done by private organizations (27). Hence, the private sector interest in CTI has grown throughout time. Businesses rely more on digital systems and, therefore, expand their attack surface. CTI empowers organizations to defend against emerging threats and adapt to the evolving cyberthreat landscape.

The sources of CTI include both private and open-source vendors' threat reports. Threat information is shared by industry, academic, and private research peers to counter the evolving cyberthreat scenario. This collaborative effort is essential to tackle the imbalance of defense against adversary attacks (33) that

range from phishing to sophisticated state-sponsored cyber espionage (34). While information sharing is important, efficient sharing is a challenge. Private vendors may offer feeds with structured information to be readily consumed, but many open-source reports are published in different formats with unstructured data (18). Many standards for sharing have been proposed to address this problem.

2.2 CTI STANDARDS AND FRAMEWORKS

The creation of standards aims to enhance interoperability between systems to increase the efficiency of knowledge sharing. Timely intelligence is a must from the defense perspective; however, it can be hindered if communication is not done properly.

Many standards have been created to tackle different problems in CTI, reason they have different usage and semantics. We will further cover relevant standards in the domain and their uses.

2.2.1 Cyber Kill Chain

The Cyber Kill Chain (35) is a phase-based attack model developed by Lockheed Martin. It proposed a seven-step model that initially described intrusion attacks as a chain of activities. This kill chain is how adversaries proceed in order to achieve the intended impact. The disruption of any of the steps would interrupt the entire process, hence achieving the defense objective.

The model is a risk management approach that gathers information about adversaries, their capabilities, and objectives to address the cyberthreat risk and support decisions regarding the organization's defense. It allows an organization to better allocate resources considering its knowledge of the adversary.

The Cyber Kill Chain is defined as the seven steps: reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objectives. The analysis of the kill chain can drive the defensive courses of action of the attacked organization.

2.2.2 Diamond Model

The Diamond Model (3) is a formal framework of intrusion analysis. The model is complementary to the Cyber Kill Chain, broadening the view of an intrusion to external relationships. It is generally represented by its Diamond Event graphic depiction in Figure 2.3 that can be described as an *adversary* deploying a *capability* over *infrastructure* against a *victim* to produce a result.

The main importance of this model is that it provides a generic and flexible, yet formal, structure to encompass the essential concepts of intrusion analysis. It does not provide an ontology, but it provides the fundamentals for one.

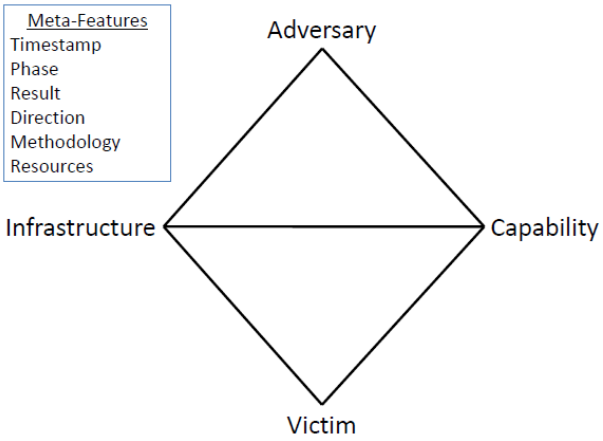


Figure 2.3: The Diamond Model and its core features and relationships (3).

2.2.3 MITRE Standards, Languages and Frameworks

MITRE is a not-for-profit organization based in the United States of America that develops technical guides in cybersecurity and other areas. It has fostered many standards, knowledge bases, languages, and frameworks to support diverse cybersecurity domains. The first one to be described is MITRE ATT&CK.

Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) (36) is a public knowledge base developed by MITRE to document and categorize Tactics, Techniques, and Procedures (TTP) of real-world adversaries. It provides a structured framework for analyzing cyber threats, enabling security teams to perform threat modeling, detection development, incident response, and consistent communication, within other activities. Similar to the Cyber Kill Chain, the framework is structured to reflect the adversary's attack life cycle as observed in the data.

ATT&CK is a behavioral model, where the tactics are the adversary's goals during the attack, the techniques are the means the adversary uses to achieve the goals, and the procedures are specific implementations of techniques as they were used by adversaries. It is divided into three technology domains: Enterprise, which includes traditional networks and cloud systems, Mobile, and ICS, for industrial control systems (36). In Enterprise version 17.1, there are 14 tactics and 245 techniques described in ATT&CK. The tactics are Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact.

Beyond describing TTPs, ATT&CK has a repository with report examples with its usage, description of common Threat Actors, malware, and tools, mitigation, and other resources. The knowledge base is widely used across organizations.

Malware Attribute Enumeration and Characterization (MAEC™) (37) is a standardized language developed by MITRE to describe malware and its analysis. It provides a structured format to share information about malware attributes, behaviors, and impacts. This information can be leveraged to enhance detection, correlation, and response to incidents involving malware. As part of MITRE's environment, it enables interoperability among threat intelligence platforms as well as integration with other standards such as STIX and CybOX.

Common Attack Pattern Enumerations and Classifications (CAPEC™) (38) is a publicly available catalog developed by MITRE of known attack patterns used by adversaries. By sharing this kind of information, the catalog allows cybersecurity analysts to understand approaches taken by adversaries and how the weakness of applications may be exploited. Capturing attack patterns is an important step in cyberthreat analysis as it represents how an adversary exploits. It relates to TTP as the latter describes a high-level goal from the entire attack life cycle.

Each entry has a description of the attack mechanisms, a severity classification, relationships with attack patterns, prerequisites for its occurrence, resources required, and references to other frameworks as ATT&CK, CWE, and CVE. The usage of CAPEC empowers threat modeling and defensive planning, allowing cybersecurity professionals to assess vulnerabilities and develop mitigation strategies.

Cyber Observable eXpression (CybOX™) (39) has been a standardized language developed by MITRE to represent cyber observable events, which are properties that can be detected in an operational cybersecurity context. It enables a structured way of characterizing low-level technical details, as file hashes, domains, IP instances, and network connections, among others, to be automatically shared across tools and systems. Its usage is complementary to broader standards by comprehending fine-grained details over observed events and artifacts. The standard was later integrated into STIX in version 2.0.

2.2.4 STIX

Structured Threat Information Expression (STIX) (8) is a standardized language developed by MITRE and maintained by OASIS for representing and sharing cyber threat intelligence (CTI) in a machine-readable format. It is largely adopted by cybersecurity tools and platforms, which allows an effective collaboration among organizations and sectors (40). STIX encompasses a set of cyberthreat concepts to enable a detailed description of those threats, including TTP, Indicators of Compromise (IoC), and other technical features related to adversaries, as well as a broader view of campaigns, incident reports, and the relationship between Threat Actors.

STIX language is comprehensive and extensible, so it can comprise complex data on threats. It consists of three main objects as part of its STIX Core Objects: STIX Domain Objects (SDO), STIX Cyber-Observables Objects (SCO), and STIX Relationship Objects (SRO). SDOs are Higher Level Intelligence Objects, which are common concepts an analyst would need to describe a CTI. SCO comprises the former CybOX language to describe technical details. SRO represents any relation between two STIX objects.

There are 19 classes of SDO defined. Their descriptions are in Table 2.1. For SCO, there are 19 classes defined: Artifact, Autonomous System, Directory, Domain Name, Email Address, Email Message, File, IPv4 Address, IPv6 Address, MAC Address, Mutex, Network Traffic, Process, Software, URL, User Account, Windows Registry Key, Windows Registry Value Type, and X.509 Certificate. Relationship objects, SRO, have two defined objects: Relationship and Sighting. Each SDO and SCO has a list of relationships listed in the object definition; the user may use one of them or any user-defined relation.

One straightforward usage of STIX is to create knowledge graphs of CTI. In version 2.1, STIX uses JSON notation to facilitate the consumption of the information directly by systems. SDO and SCO can represent nodes connected by SRO, which can be the edges. Figure 2.4 depicts a fictional example of a mal-

Table 2.1: STIX Domain Objects and their Descriptions, adapted from (8)

STIX Domain Object	**Description**
Attack Pattern	A type of TTP (Tactics, Techniques, and Procedures) describing ways adversaries attempt to compromise targets.
Campaign	A grouping of adversary behavior describing its malicious activities over a period against a specific set of targets.
Course of Action	A recommendation or action to mitigate, respond to, or prevent threats.
Grouping	A collection of STIX Objects with a shared context, but without requiring a relationship between them.
Identity	A characterizing object that defines individuals, organizations, systems, or groups.
Incident	Yet to be defined by OASIS.
Indicator	A pattern used to detect suspicious or malicious activity. It may be defined in terms of SCO.
Infrastructure	It describes systems, services, or any physical or virtual resources used by threat actors.
Intrusion Set	A grouped set of adversary behavior and resources used repeatedly across multiple attacks believed to belong to a single organization. It is a superset of campaigns and activities that indicate a Threat Actor.
Location	A specific physical or geopolitical location.
Malware	A type of TTP describing malicious code, should it be a program, a payload, or another.
Malware Analysis	A specific assessment of malware samples, providing details on their behavior, capabilities, and indicators.
Note	A textual annotation to provide context that cannot be represented by STIX Objects.
Observed Data	Represents artifacts related to cybersecurity entities using SCO.
Opinion	An evaluation of the information that was produced by another entity.
Report	A set of CTI describing the details and context of one or more topics.
Threat Actor	An individual, group, or organization conducting malicious activities.
Tool	A legitimate software, script, or utility that can be used in cyberattacks.
Vulnerability	A weakness in software or hardware can be exploited for malicious means.

ware analysis using an OASIS STIX visualization tool [1]. This example has two SDO – a malware analysis and a malware, named "Guildma payload" – and two SCO – a file, named "guildma.exe", and a directory. The relationships among them are described in the edge. The source code of the STIX bundle depicted in Figure 2.4 is in the appendix. The knowledge graph representation helps both machine processing and analyst comprehension.

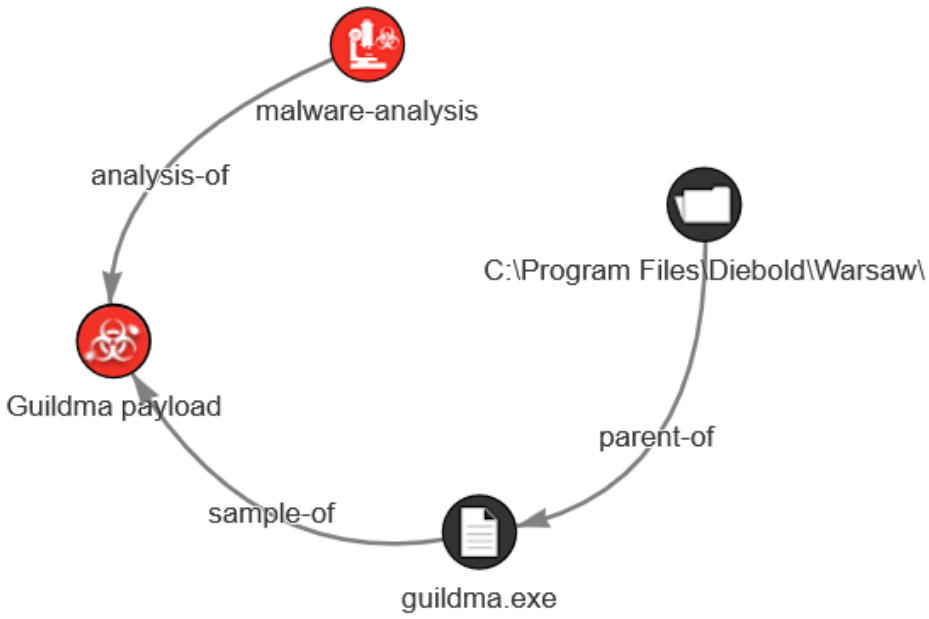


Figure 2.4: A STIX bundle of a fictitious malware analysis.

While there is a set of pre-defined objects, STIX is extensible, so it is possible to describe any context. For instance, if one is to describe an attack using Bluetooth channel, they may define an SCO for the Bluetooth Address using STIX Extension Definition notation using any needed property. An extension example can be seen in Listing 2.1.

Listing 2.1: Extension Definition for x-bluetooth-address SCO

```
{
  "id": "extension-definition--ba6d32f9-3f99-4cc5-a779-af118b424777",
  "type": "extension-definition",
  "spec_version": "2.1",
  "name": "Bluetooth Address SCO Extension",
  "created": "2025-05-25T12:00:00.000Z",
  "description": "Custom SCO to represent Bluetooth MAC addresses",
  "extension_types": ["new-sco"],
  "schema": "https://example.com/schema/x-bluetooth-address/v1/",
  "version": "1.0",
  "created_by_ref": "identity--e5f87884-f08c-440e-b9e3-9e3819f292f8"
}
```

[1] <https://oasis-open.github.io/cti-stix-visualization/>

2.3 NATURAL LANGUAGE PROCESSING

The field of information retrieval has significantly advanced in NLP with the evolution of techniques, from traditional text analysis like Term Frequency-Inverse Document Frequency (TF-IDF) and bag-of-words to Deep Neural Networks (DNNs) (9). The beginning traces back to machine translation research that sought to enhance translation between two languages using the capacity of modern computers (41). The possibility of it has shown to be promising, though challenges have arisen. This section will go through the adoption of NLP in cybersecurity and its techniques.

The initial approaches primarily relied on rule-based systems and statistical models. They were limited in usage, and the calculation could be computationally intensive. Rule-based approaches set handcrafted patterns and regular expressions to retrieve characteristics and known entities in text. Statistical methods use probability theory and statistical models to analyze text, such as N-gram models and TF-IDF. The first approach is deterministic and more dependent on the knowledge about the languages, while the second is data-driven and adaptive, which makes it more flexible for NLP.

The area delved into different applications than translation, such as document classification and entity extraction, using known algorithms. Researchers have applied classical machine learning classifiers such as Naive Bayes (NB) and Support Vector Machine (SVM) to classify documents, but their performance depends on domain-specific feature extraction. Feature engineering would require significant manual effort, setting constraints for scalable use. The quality and relevance of the features directly affect the model's performance. Thus, when computing capacity increased, deep learning methods established a new standard, leading to more flexible and scalable models.

2.3.1 Deep Neural Networks

Previous statistical methods struggle with long-range dependencies, rely on manual feature engineering, and fail to adapt meanings to context. Deep Neural Networks (DNN) address these issues by learning patterns and keeping memory in long contexts. Subsequent approaches based on neural network architectures such as Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTM), and Convolutional Neural Networks (CNNs) established a new standard for NER.

The simplest type of neural network is a *perceptron*, a single layer of weights applied to input features that can be represented as

$$y = \mathbf{w}^T \mathbf{x}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]$ and $\mathbf{x} = [1, x_1, \ldots, x_d]$ are augmented vectors of weights and inputs, respectively taking in consideration the bias weight and d is the number of features of the model (4). When an activation function f is applied to it, it defines a hyperplane dividing the input space into two and creating a binary classifier. This paves the notion for DNN, which are sets of parallel perceptrons stacked into interconnected layers. While a single layer of perceptrons can approximate linear functions, multilayer

perceptrons can implement nonlinear discriminants. When the network has more than one hidden layer – the layers that are neither the input nor the output –, it is called **deep neural network**.

Neural networks have specialized variants that excel in certain problem domains. Convolutional neural networks (CNN) take advantage of data that have a known topology with local structure (4), where closer elements are more strongly related than further ones. While it is common in image processing, where the data is commonly a matrix, CNN can be applied to NLP due to the relation among close words in a sentence, being useful for classifying documents.

Recurrent neural networks (RNNs) are specialized to handle sequential data. In this kind of DNN, perceptrons do not have the constraint of being parallel, so one input can be dependent on another one. This creates a state for a hidden unit, acting as a form of memory (4).

In NLP, the input data is a text to be processed, and these deep learning models process text word by word in a left-to-right manner, mimicking human reading, using past information to predict the next one. An RNN can retain information for as many states as one layer has layers. A way of understanding it is to expand it as a front-forward equivalent neural network, such as in Figure 2.5

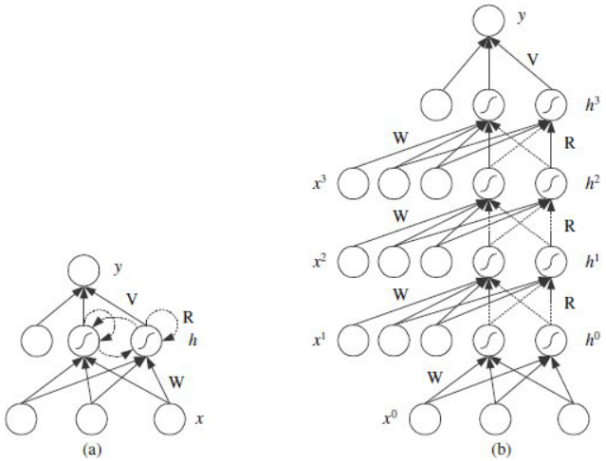


Figure 2.5: In the left, a Recurrent Neural Network; in the right, its equivalent expanded neural network (4)

RNNs still struggle with longer context due to the vanishing gradient problem (42). Long Short-Term Memory (LSTM) networks are a type of RNN that improves this by introducing memory mechanisms to retain information over long sequences. The LSTM introduces memory cells and gating mechanisms to the network. Memory cells allow the information to be kept over time unless it is explicitly altered. Gates are sigmoid layers that work as switches to control which information will be altered or not. Those functions enhance dealing with long-term dependencies and contextual meaning.

CNN, RNN, and LSTM are fundamental models for NLP. Along with them, text embeddings developed as well, taking advantage of advances in neural networks. Embeddings are a core part of NLP, so unstructured text can be processed by models.

2.3.2 Text Embeddings

Embedding refers to the representation of text elements in a continuous vector space. It maps tokens, which can be words or parts of them, into dense vectors that capture the semantic meaning and relationships. The activity of retrieving the tokens is called tokenization.

Tokenization is the process of breaking down raw text into smaller, meaningful units called tokens. These tokens can be words, subwords, characters, or even symbols, depending on the tokenization strategy. It is a preprocessing step that transforms unstructured text into a structured format suitable for computational models.

Since natural language is inherently ambiguous and unstructured, tokenization bridges the gap between raw text and numerical input needed in machine learning models. Once a text is tokenized, each token is converted into a numerical vector before it is processed, may it be by statistical methods or neural networks. The embedding is, therefore, applied to the token, mapping it to a dense vector.

Word2Vec (43) is an embedding technique based on a shallow neural network trained to predict word contexts in large text corpora. The embedding groups words with similar meaning close together in the vector space, capturing the semantic similarity among words. It has set a groundbreaking technique, because it permitted generalization across words based on the learned similarity. Even so, this model does not adapt the meaning of a word in different contexts since each word is assigned only one vector. Word2Vec faces challenges with polysemy.

GloVe (44) and **FastText** (45) are embeddings that brought innovations and enhancements, yet they still apply static vectors and do not handle the meaning of words in context adequately. Embeddings from Language Models **(ELMo)** (46) leveraged bidirectional LSTM (Bi-LSTM) to generate a contextualized word embedding. It improves the ability to understand meanings in context. ELMo employs two separate LSTM networks, one that processes left-to-right and another that does it right-to-left. ELMo generates context-sensitive embeddings that capture the meaning of a word based on its usage within a sentence. It does so by computing the weighted combination of all hidden layers from the BiLSTM.

Contextual embeddings have set a pioneering standard in the field of NLP. Researchers leveraged neural networks' capacities and developed strategies that can tackle the polysemy problem. Considering the fundamental problem of machine translation in NLP, these embeddings are used to map texts of an input language to an encoder, which transmits the context vector to the decoder, and is then transformed into the output language using the target embedding. Following models modified the basic encoder-decoder architecture to improve the performance in tasks such as text generation or text classification by using only one part of it. The large language model (LLM) Bidirectional Encoder Representations from Transformers is one of these, and it is used as an embedding for state-of-the-art solutions, but one has to understand the Transformers first.

2.3.3 Transformers

Transformers (5) is a model architecture based on the attention mechanism. It employs a neural network without recurrence or convolution, differentiating from RNN and CNN. Instead, it relies solely on the

attention mechanism, which is a method to capture the relation between the parts of the input and the parts of the output. It allows assigning different weights to each input element depending on the current token, focusing on the elements that are more important for each element.

When the attention relates the input to itself, it is called *self-attention mechanism*. Each element of the sequence attends to other elements within the same sequence, so an element is computed among all the elements at the same time. It helps to capture the semantics of a term within the specific context of the sequence, improving in comparison to left-to-right and right-to-left methods. By doing this, it captures dependencies regardless of the position and distance of the element. This strategy handles long-term dependencies more effectively while capturing the contextual semantics.

The attention function can be defined in terms of its core elements: a query Q, a key K, and a value V, which are vectors typically obtained via learned linear projections. These vectors contain the adaptable weights for each input element to set the context vector. Given a sequence $X = [x_1, x_2, ..., x_n]$, for $X \in \mathbb{R}^{n \times d}$, the attention function is

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where d is the input dimension and d_k is the dimension of the key and query vectors. As it is seen, it is used a dot-product attention with a scaling factor. The application of the softmax function ensures that the weights are normalized.

The self-attention in the Transformer architecture improves not only the capturing of dependencies and the contextual representation, but also computing performance. The dot-product operation on vectors allows parallel computation across tokens, which improves training speed and scalability of models. The Transformer uses a multi-head attention, which are linear projections of the attention function that are later concatenated. Each projection is independent and computed in parallel. Once the paradigm shifted from sequential to attention-based parallel processing, NLP achieved an unprecedented performance, shaping future modeling.

The Transformer architecture consists of 6 layers of encoders and decoders stacked. The encoder has one self-attention sub-layer and one position-wise feed-forward sub-layer. The decoder has one masked self-attention sub-layer, one attention sub-layer connected with the encoder, and one position-wise feed-forward network sub-layer. One layer of encoder-decoder is depicted in Figure 2.6.

To better understand how the self-attention works in Transformers, Figure 2.7 shows the output of the last layer of encoders in relation to the word “it” in the sentence “The animal didn’t cross the street because it was too tired”. It can be seen that the attention mechanism focuses more on the words “The animal” for the encoding, reflecting the objective of capturing the relation among terms.

“Attention is all you need” proposed the Transformer architecture for machine translation tasks, but it has paved the use in other modalities of input and output. On language representation models, the Transformer has been adapted to leverage the encoder capability.

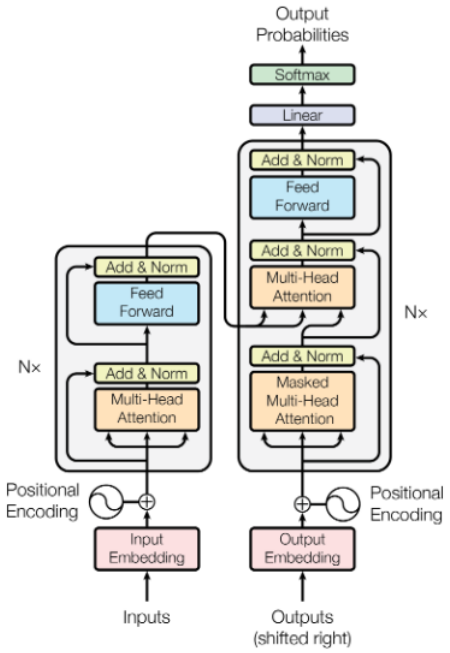


Figure 2.6: The illustration of the sub-layers of encoders and decoders (5).

2.3.4 BERT

Bidirectional Encoder Representations from Transformers (BERT) (7) is a language representation model pre-trained on deep bidirectional representations. The model is designed to improve fine-tuning, reducing the need to adapt the model to tasks, which makes it a more general approach. It allows fine-tuning to a range of tasks, only training a single output layer from the start, reducing the amount of modifications on parameters, thus reducing training cost.

BERT uses an adapted version of Transformers, removing the decoder part. This choice fundamentally differentiates it from GPT (47), which uses the decoder. While the decoder keeps attention in an auto-regressive, left-to-right manner as previous approaches of DNN, the encoder attends to all tokens with bidirectional context. GPT architecture tackles text generation problems, justifying its preference for the decoder; meanwhile, BERT addresses semantic understanding for tasks such as text classification, question answering (QA), named entity recognition (NER), and others.

The original paper defines two versions of BERT that specify three variables: the number of layers (L), the hidden size (H), and the number of self-attention heads (A). The layers are Transformer encoder blocks. The hidden size is the dimension of the output, that is, the dimension of the embedding. The self-attention head count refers to the multi-head attention sub-layer in the Transformer encoder block. BERT models' sizes are:

$$\text{BERT}_{\text{BASE}} : \text{(L=12, H=768, A=12, Total Parameters=110M)}$$
$$\text{BERT}_{\text{LARGE}} : \text{(L=24, H=1024, A=16, Total Parameters=340M)}$$

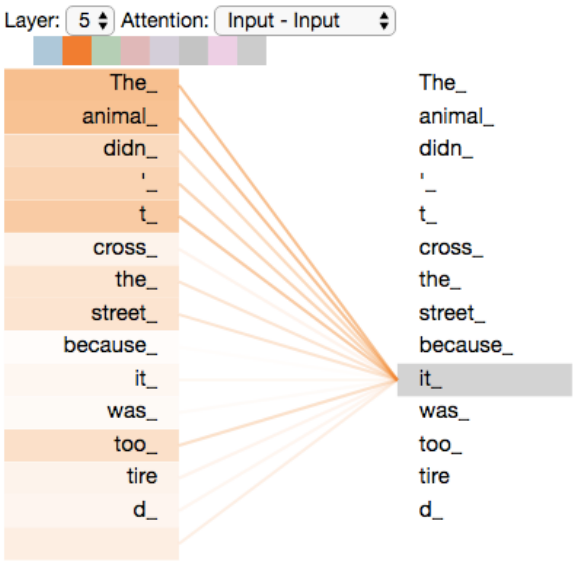


Figure 2.7: Self-attention mechanism focus in relation to the word “it” (6).

BERT is designed to handle both single sentences and pairs of sentences in a single sequence. To achieve it, it uses two special tokens: a classification token (`[CLS]`) that starts all sequences and is used in classification tasks, and a separator token (`[SEP]`) that ends all sentences, therefore splitting two sentences when present. These tokens are part of the token embedding, which is the WordPiece embedding, with a vocabulary of 30,000 tokens. Along with it, BERT employs one embedding that indicates to which sentence one token belongs, and one embedding to indicate the position of the token. The final representation is the sum of the three embeddings, resulting in the input embedding.

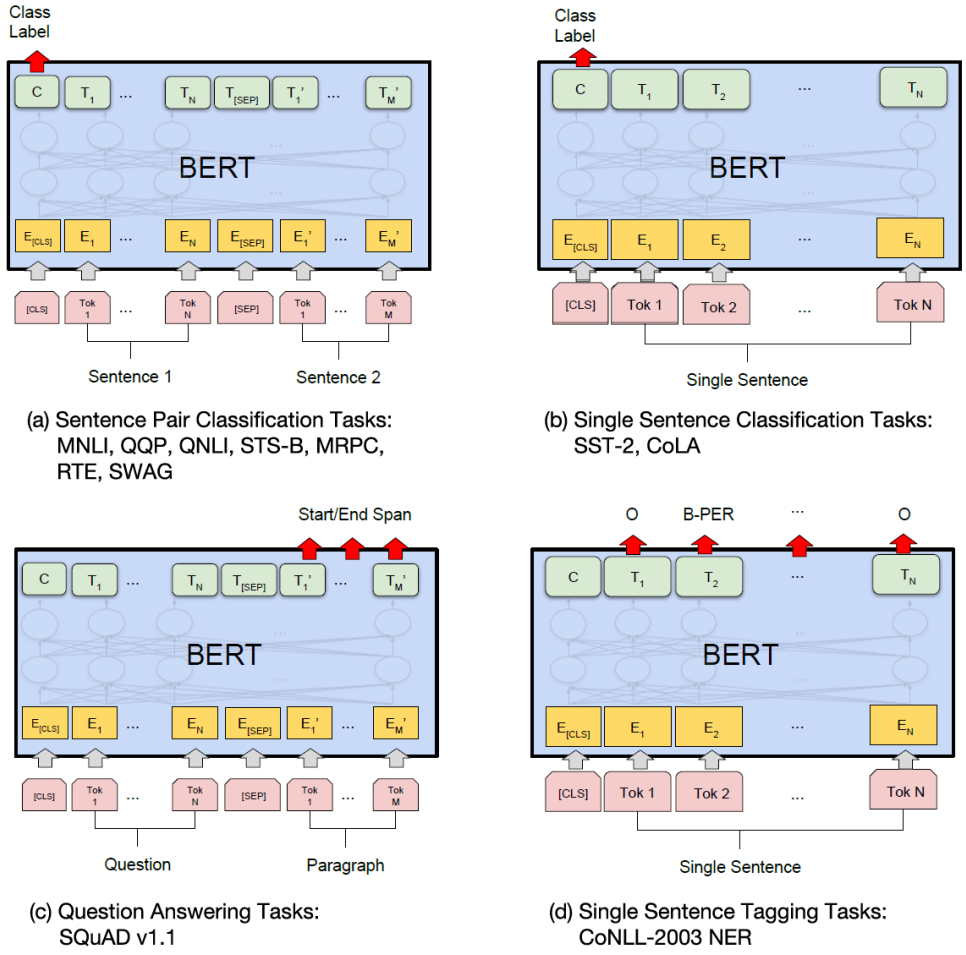
The model introduces an alternative pre-training objective. It employs "masked language model" (MLM), based on Cloze task (48), which requires filling the blank on a sentence with the word that best aligns with the context, as well as "next sentence prediction" (NSP), which assesses whether one sentence logically follows another in a text.

For the masked language model task, unlabeled texts are selected, and 15% of the tokens are masked at random. They are replaced by the special mask token (`[MASK]`) and the model has to predict this token. Because this token does not appear when fine-tuning, the masking strategy to mitigate this problem is to actually mask the selected token 80% of the time, and keep it unchanged or replace it with any random token the rest of the time. With this task, BERT learns the context from all the tokens.

The following pre-training task is next sentence prediction. It is generated from unlabeled texts, dividing them into two spans of text. For the training, 50% of the time the sentence will be the actual next sentence, while the remaining time it will be substituted by a random sentence from the corpus. This training aims to enhance the model’s understanding of relationships between sentences for tasks with a similar arrangement, such as QA.

Fine-tuning the model will update all parameters end-to-end. The difference from task to task will be adjusted on inputs and outputs. For sequence tagging tasks, for instance, the output tokens are fed into an output layer, which will outcome the resulting vectors. Classification problems’ results are output in the `[CLS]` representation.

The original model paper presented the results of fine-tuning in 11 NLP tasks, which will not be extensively covered. For Named Entity Recognition, fine-tuning BERT expects to input one sentence and output the final hidden representation to a classification head, as seen in Figure 2.8. The model is fine-tuned in CoNLL-2003 NER (7); the details can be seen in the preprint version (49). The classification layer is non-autoregressive and does not use Conditional Random Fields (CRF). WordPiece cased version is used for embedding, and only the first subtoken of a word is used for prediction.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 2.8: Arrangement of input and output of fine-tuning BERT on different tasks (7).

Robustly optimized BERT approach (RoBERTa) (50) proposes changes in BERT's pre-training and text encoding while using the same core architecture. RoBERTa finds that the original model is undertrained, for which it considers a larger set of text corpora. Moreover, the size of training batches, the size of sequences, and the masking strategy are modified. Optimization parameters are mostly unchanged. The training objectives are also modified by removing NSP since RoBERTa experiments achieve better results having NSP loss removed. Lastly, the input representation is scrutinized, choosing GPT-2 (51) Byte-Pair Encoding (BPE) adaptation. BPE uses a byte-level representation rather than Unicode to avoid increasing the vocabulary size and still deals better with out-of-vocabulary words than word-level representation. With these adjustments, RoBERTa outperforms the original BERT in tested benchmarks.

One important accomplishment of BERT and the following models based on it is their transfer learning capacity. Pre-training with a self-supervised strategy in large corpora allows models to acquire general

knowledge of the language, its structures, patterns, and intrinsic meaning. This is quite useful for transferring knowledge in downstream tasks. Furthermore, domain-adaptive pre-training (DAPT) (52), in which further self-supervised training is performed in a domain-specific corpus, takes advantage of knowledge transfer in low-resource domains. That has been demonstrated in biomedical (15), financial (16) and cybersecurity (17) fields.

BERT architecture established a foundation on which many models were developed. Along with it, the transformers also enhanced autoregressive models, especially in the text generation task, as seen in GPT and GPT-2. Research has shown that these models do perform well in named entity recognition tasks, even without fine-tuning (53). As models scaled up, for instance, as seen in PaLM (54), which has 540B parameters, so did their capacity. This is further exploited in multimodal models, in which inputs different from text are also used.

Multimodal models train on diverse datasets, enhancing knowledge variety. These models exhibit great results in many tasks, including NLP. One family of models that sets the state-of-the-art in language modeling is Gemini.

2.3.5 Gemini

Gemini (55) is a family of generative multimodal models. The models are trained jointly in datasets of image, audio, video, and text to improve general capability and understanding and reasoning performance. It leverages Transformer decoders with improvements in the architecture, such as multi-query attention.

The models are natively multimodal, meaning that training is performed in mixed data and multilingual. They can handle interleaved sequences of different input types while outputting as both image and text. Three different sizes of models were made available: Ultra, the largest of them, targeting the capability to deal with high complexity tasks; Pro, a version optimized for performance in terms of cost; and Nano, designed for running in low-resource devices by distilling larger Gemini models. The smallest model, Nano-1, has 1.8B parameters. All models use the SentencePiece tokenizer.

For this study, we will consider only its text-to-text capability, since we are interested in its application for NER. Gemini is evaluated in many text benchmarks, such as MMLU (56), which is a multitask consisting of 57 subjects split into four categories: humanities; social science; science, technology, engineering, and mathematics (STEM); and other, covering a broad range of knowledge domains. The results in text domains show indirectly promising capabilities in named entity recognition.

Gemini 1.5 (57) scaled the long-context from the previous version. It introduces two versions: Gemini 1.5 Pro and Gemini 1.5 Flash. Its main breakthrough is augmenting the context windows from 32k tokens to 10M tokens in the Pro version. The models have a mixture-of-experts (MoE) architecture with a learned routing function to select the path to be taken by the input. The 1.5 Flash model performs better than the 1.0 Pro model in most benchmarks, even though it is smaller and more efficient than the 1.0 Pro version.

Autoregressive models generate coherent output, i.e., coherent text. Despite not being initially developed for classification problems, as discriminative models are, the enhancement in capability showed that they perform quite well by adapting their usage. One strategy is to fit the objective by reformulating

the task in the prompt. Setting the NER task as a natural language generation problem allows the model to classify the input (58). The model ingests a prompt – the user input – with instructions to shape the problem. This instruction fine-tuning will be further exploited.

2.4 NAMED ENTITY RECOGNITION IN CYBERSECURITY

In this section, we review studies related to the Named Entity Recognition task performed in cybersecurity. Methods used range from traditional NLP techniques to LLM. The cybersecurity realm is wide with many subdivisions, which can be seen in the many different objectives of studies. At the end of the section, the studies are summarized.

Rahman *et al.* (13) conducted a literature review on text mining for CTI. The main objective of the research is to set a starting point for researchers on the use of mining CTI from unstructured text. It does so by looking what are the sources for CTI collection, what are the primary purposes for mining CTI, and which are the means to execute it. An initial cut found 28,484 publications from 2000 to 2020. A set of inclusion and exclusion criteria is applied, selecting 38 papers for further analysis. They only include English papers that can be publicly reached, meaning bias must be accounted for in the results.

The study points out that they found seven sources of CTI, threat reports being the primary source of CTI, representing close to 37%. These reports are technical publications from specialized sources. They contain information about the attacker's TTP and IoCs discovered, and are generally exploited for the extraction of these two features. Some present tools for automation, allowing gathering IoC in bulk. Nonetheless, hacker forums, Twitter (now named X), and security blogs are also valuable. These sources are explored in other ways, such as topic identification and mining from natural languages.

Regarding the purposes, seven types have been identified, though four of them cover most of the reasons: threat event identification, threat topic analysis, IoC extraction, and Tactics, Techniques, and Procedures (TTPs) extraction. Threat event identification and threat topic analysis are the most present purposes, which show motivation to proactively enhance defense against possible future attacks. Those are related to Twitter feeds as a source. When observing studies for IoC extraction and TTPs extraction, studies tend to use threat reports, as it is expected to have technical information related to attackers. It is noted that studies do not cover both IoC and TTP extraction jointly.

The authors arrange the approaches in more general categories, in which they found that the most frequent are semantic relationships among CTI keywords, keyword identification, and topic classification. These are related to NLP because the sources are from unstructured text in general.

TTPDrill (59) is one of the studies analyzed by Rahman *et al.* that developed a tool for the extraction of TTPs. It focused on creating a threat-action ontology and extracting structured TTP from unstructured reports. Given the initial lack of a machine-readable format for TTPs, the authors introduced an ontology based on MITRE ATT&CK and CAPEC frameworks, comprising seven distinct classes.

The methodology involved a multi-step approach. Initially, a scraper was developed, and an SVM classifier was trained to distinguish relevant security articles from advertisements. The classifier utilized three

features: the article's word count, the density of security-related verbs, and the number of security-related nouns. Before NLP processing, common objects identified by a set of predefined regular expressions were substituted with a generic name. Later, these names were replaced with the original term. The text was then parsed to identify the dependencies, and the model used a TF-IDF with bm25 model to map the candidates into threat actions, structuring them into STIX format for sharing. Only known attack patterns and techniques are kept, so they can be mapped into STIX 2.1 format.

TTPDrill processed 17,000 Symantec reports and extracted threat information from them. The authors used 30 reports to set the thresholds of similarity by establishing an empirical value from the observation of accuracy metrics. Later, 50 reports were manually annotated as ground truth to measure the model's effectiveness. The work is a valuable reference for addressing the automation of TTP extraction.

Research in NLP applied to cybersecurity also leveraged traditional techniques to find correlations between security named entities. An RDF-CRF model was proposed by Yi *et al.* (23) to extract security named entities using regular expressions and a known-entity dictionary to improve performance. The regular expressions composed a rule-based extractor to locate specific entities, while the dictionary-based extractor utilized known-entity lists. Afterwards, the study employed the CRF extractor on selected features to exploit the correlation between those entities and improve the recognition performance.

The authors acknowledged the scarcity of large publicly available datasets for extracting cybersecurity named entities. To tackle this limitation, they constructed a dataset by collecting data from security forums, vendor bulletins, and blog articles. This dataset was manually annotated using a majority vote mechanism from the analysis of three annotators. This made up the ground truth dataset of 14,000 marked texts. There are 26 entities arbitrarily selected using BMESO notation.

The effectiveness of the RDF-CRF model was compared in two ways. Firstly, it was analyzed together with the statistical models Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM), and CRF itself. The evaluation is divided per entity classes, and the CRF model achieved an overall high performance. Following it, the effectiveness of the model is evaluated by comparing it to two state-of-the-art methods at the time. The experiment evaluated an LSTM-CRF model and an FT-CNN-BiLSTM-CRF model, as neural network methods often have high performance in complex tasks. The RDF-CRF model was able to outperform these baselines.

TIMiner (11) is a framework for extracting CTI from social media data. The study addresses two challenges: the automated extraction of IoC and the classification of CTI in domain tags from social media. TIMiner aims to enable personalized CTI sharing to five different domains, which are Industrial Control System (ICS), Internet-of-Things (IoC), finance, government, and education. The authors developed a CNN-based domain recognizer over text spans embedded with Word2Vec (43) model.

The authors collected threat-related descriptions from various social media. They developed TI_spider, a system of distributed crawlers that monitored 75 vendor bulletins, collecting more than 118,000 threat-related descriptions from 2002 to 2018. Five researchers manually labeled 15,000 threat descriptions for the final dataset.

The method for hierarchical IoC adopted a comprehensive strategy to effectively identify the terms. At first, indicators with a well-known structure are retrieved with regular expressions. Following it, a

BiLSTM+CRF model is employed for NER. Lastly, a top-5 word similarity algorithm identified unknown IoCs, taking into consideration the consistently growing number of these entities.

Sauerweinand and Pfohl's systematic review (9) aimed to identify approaches of NLP and ML to extract TTP from unstructured text and evaluate which methods are more effective to accomplish the task. The study took papers from 2009 to 2020, yielding an initial count of 5,307 papers. After the selection of papers that were useful to address the research questions, 26 of them met the criteria. These papers were evaluated regarding two characteristics: which NLP techniques were used and which ML model was used.

The study found that Support Vector Machine (SVM) and Naive Bayes (NB) were the most frequent machine learning models for classification tasks. These methods were preceded by traditional NLP processing of tokenization, POS tagging, IoC replacement, lemmatization, one-hot encoding, and binary relevance transformation.

The authors noted the lack of training datasets for the evaluation of these techniques and highlighted the need for publicly available datasets as baselines. They generated one dataset from the MITRE ATT&CK knowledge base, as manual labeling was deemed to be too time-consuming. It was observed that the dataset was imbalanced, posing a challenge for evaluation.

TCENet (Threat Context Enhanced Network) was the classifier of the framework TIM (TTPs Intelligence Mining) (60). It proposed a pipeline of extraction, embedding, classification, and exportation for two intelligence products: a Sigma attack detection and a STIX Bundle.

The study incorporated the Transformer-based model BERT in its description feature embedding. SentenceBERT was employed in the TCENet model. This model consisted of two neural networks: one CNN and one BiLSTM with an attention mechanism. The dataset was built from crawling 10,761 reports of 5 security vendor blogs. The overall accuracy was based on six TTPs, and it outperformed the BERT-CLS model, though it did not specify if it was the base or large version. The authors indicated that pre-trained models perform better than static word embedding.

In the study conducted by Alves *et al.* (61), eleven BERT models were evaluated on classifying sentences according to the corresponding MITRE technique. The models were variations of BERT, RoBERTa, DistilBERT, and DistilRoBERTa, both cased and uncased. It was also considered the models SecBERT and SecRoBERTa, which are addressed later.

The dataset was MITRE's public repository of examples. The data were sentences that illustrated one technique. There were 466 out of 576 techniques exemplified in the dataset. The authors established a minimum count of five sentences to be considered representative. Based on this, 9909 sentences were selected, comprising 253 techniques and sub-techniques. The study also observed the imbalance of the dataset, as the class with the most examples had 371 instances, while the least had only five.

Among the models, RoBERTa Large and BERT Large performed better when using all 253 techniques and sub-techniques. Though TCENet shows better accuracy, it only uses a subset of the classes.

The adaptation of BERT models for a specific domain is generally beneficial, it is reported by FinBERT (16), which is tailored for the financial domain. The exposure of models to the domain-specific texts, such as financial, may alter the advantage of pretraining.

FinBERT evaluated models both with and without domain-specific vocabulary, and the model with base general vocabulary achieved the best result in almost all datasets. It's also been noted that training BERT-like models on small datasets can lead to instability in their performance.

The result shown in FinBERT enforces the possible results in the cybersecurity domain. This approach was explored in many studies, as well as fine-tuning for specific tasks. As there are many different subdomains in cybersecurity, this adaptation may vary. Studies investigating tailored versions for cybersecurity are further explored.

RuCyBERT (62) explores the fine-tuning of BERT models in the cybersecurity domain in the Russian language. The study compares the quality of NER using multilingual BERT, RuBERT – a Russian-tuned model –, and RuCyBERT – a fine-tuning of RuBERT in cybersecurity reports. RuCyBERT outperforms the others in classifying cybersecurity domain entities, but RuBERT performs better on generic entities like location and organization. The authors also present a new version of the Sec_col corpus, augmented with 861 unstructured texts related to the domain.

Ameri *et al.*'s CyBERT (20) classifies cybersecurity feature claims from ICS based on its documentation. CyBERT shows better accuracy in this task compared to ELMo and ULMFiT models. The authors created the training set from 19,793 documents, which resulted in circa 215,000 sentences. The classifier was trained on 6,763 sentences.

Ranade *et al.*'s CyBERT (12) is a domain-adapted BERT on a large cybersecurity corpus with an extended vocabulary to include domain-specific words. CyBERT was downstreamed into multi-class classification and NER tasks. The authors noted the lack of large-scale public cybersecurity texts for NER, and three annotators manually labeled a dataset with 66% agreement to determine the most accurate labels. Labels were a set of seven entities. The model was compared to BERT only qualitatively, indicating CyBERT produced more suitable responses.

SecBERT[2] and SecRoBERTa[3] are models stated to be pre-trained on cybersecurity text sources: APTnotes, Stucco-Data, CASIE, and SemEval-2018 Task 8. Its vocabulary has been adapted to the corpora. Though it does not have any publications regarding the training method and evaluations, its popularity on Hugging Face, an open-source platform for Machine Learning, caught our attention, and it was further evaluated in this study.

SecureBERT (17) is a RoBERTa model focused on CTI and automation. The choice of a smaller model is due to its lower training costs. The author claims that the cybersecurity domain has less diversity than general language, so a smaller model should suffice for the task. The dataset for training has 1.1 billion words from many sources. Two versions of SecureBERT are evaluated on NER using MalwareTextDB (63), showing better performance than RoBERTa-base and SciBERT.

STIXnet (21) is a modular framework to extract entities and relationships in STIX format. After extracting the text from the input, the pipeline first extracts the entities. It processes the input on three different submodules: IOC Finder, which is based on regular expressions; Knowledge Base (KB), which has a database from a private company over years of manual annotation; and rcATT (64) for recognizing TTPs.

[2]<https://huggingface.co/jackaduma/SecBERT>
[3]<https://huggingface.co/jackaduma/SecRoBERTa>

After merging the results, the relationships are extracted using two techniques: a rule-based approach that leverages spaCy dependency parser output to find the verb linking two entities and classify it based on its similarity to an SRO, and an approach based on SentenceBERT embedding, which calculates the cosine similarity to STIX relationships embeddings. The dataset employed for evaluation was self-generated from manual annotations of descriptions of threat actors from the MITRE ATT&CK framework.

CySecBERT (19) introduces a BERT model adapted to the cybersecurity domain by further pre-training it on the domain and then fine-tuning it for three tasks: NER, general relevance classification, and CTI classification. For the pre-training phase, the authors propose the task of word similarity as a variation of MLM, where the model should predict whether two words are similar, along with a dataset clustering task. The dataset was compiled from four sub-corpora: blog data, arXiv articles, the National Vulnerability Database (NVD), and Twitter posts. It sums up to 4.3 million entries. The dataset repository includes original references and the crawler, so it's possible to reproduce it, though it may have minor deviations from the original. For NER, only three classes unrelated to CTI were searched: Software Naming, Software Version, and Attack Complexities, and the authors note the lack of datasets for the task.

KnowCTI (33) introduces a method for generating a threat intelligence graph. It employs Graph Attention Networks that are trained in both NER and relation extraction at the same time. BERT model is used as an embedding for the graph node. Five cybersecurity graduate students annotated the training data, 4,896 instances for text classification, and 8,872 instances for CTI extraction. KnowCTI was compared with four different models, both neural networks and Transformer-based, showing better performance.

The two systematic reviews analyzed provide information on the extraction of CTI until 2020. They conclude that traditional NLP techniques and Machine Learning algorithms are mostly applied to unstructured texts to extract cybersecurity entities. It draws attention that the Transformer architecture and BERT models are not present in the analyzed studies, even though their release dates are 2017 and 2019, respectively. We also analyzed studies within the scope of these goals and summarized our findings in Table 2.2. Although domain-adapted BERT models have achieved the state-of-the-art in different domains, namely SciBERT (14), BioBERT (15), and FinBERT (16) in their respective domains, the BERT model seemed to be employed in a complementary strategy in the cybersecurity domain. Therefore, we analyzed studies involving the adaptation of BERT models to cybersecurity tasks, as seen in Table 2.3.

Table 2.2: NER studies summary

Study	**Year**	**Goal**	**Data Source**	**NER Strategy**	**Dataset Availability**
TTPDrill (59)	2017	Extract TTPs from reports	17,000 Symantec reports + 50† annotated	TF-IDF + BM25	Partial
Yi et al. (23)	2020	NER from reports	14,000† security texts from CERT-CN	RDF-CRF + regex + known-entity dictionaries	No
TIMiner (11)	2020	IoC extraction and CTI categorization from social media	15,000† annotated	Regex + BiLSTM-CRF, word similarity	No
TCENet (TIM) (60)	2022	TTP classification for STIX/Sigma export	10,761† reports of 5 security vendor blogs	SentenceBERT embedding, one CNN, one BiLSTM with attention mechanism	No
Alves et al. (61)	2022	BERT variants evaluation for MITRE TTP classification	10,360 sentences from MITRE curated examples + 80† sentences	11 BERT variants	Yes
STIXnet (21)	2023	STIX NER and RE	MITRE threat actor descriptions†	Regex, Knowledge Base, rcATT (64), rule-based + SentenceBERT	Yes
KnowCTI (33)	2024	NER and RE with Graph Neural Networks	4,896† texts for classification + 8,872† texts for CTI extraction	BERT embedding, Graph Attention Network	No

†manual annotation

Table 2.3: Studies adapting the BERT model

Model	**Year**	**Training**	**Data Source**	**Training size**	**NER Downstream**
RuCyBERT (62)	2020	DAPT BERT on a Russian cybersecurity corpus with modified vocabulary	Sec_col corpus (augmented), security reports	500,000 texts	Yes
CyBERT (Ameri et al.) (20)	2021	Fine-tuning BERT on a cybersecurity feature claims	ICS device information documents	41,073,376 words	No
CyBERT (Ranade et al.) (12)	2021	DAPT BERT on a cybersecurity corpus with vocabulary extension	Security news, *CVE* vulnerability reports and *APTNotes* reports	17,000 texts	Yes
SecBERT and SecRoBERTa	2022	DAPT BERT and RoBERTa on cybersecurity corpora with vocabulary extension	APTnotes, Stucco, CASIE, SemEval 2018 Task 8	Not disclosed	No
SecureBERT (17)	2023	DAPT RoBERTa on a cybersecurity corpus with vocabulary extension	Varied cybersecurity-related text	1,072,798,637 words	Yes
CySecBERT (19)	2024	DAPT BERT on a cybersecurity corpus	Blogs, arXiv, National Vulnerability Data, Twitter	4.3 million entries	Yes

2.5 DATASETS FOR NER IN CYBERSECURITY

In this section, we analyze the landscape of datasets for named entity recognition in the cybersecurity domain. The lack of publicly available datasets was addressed in many studies (9, 20, 12, 21, 22). It is important to note that this includes different goals, for instance, the extraction of IoC and the extraction of TTP.

FEW-NERD (65) was a large-scale manually annotated dataset designed for few-shot NER. This was a general-purpose dataset; still, there were key takeaways on building multiclass datasets. FEW-NERD consisted of 188,238 sentences labeled in a hierarchy of 8 coarse-grained classes and 66 fine-grained classes. Manual annotation required extensive resources, applying 70 trained annotators and 10 experienced experts, spending close to 32 hours per annotator, with 76% agreement. The study proposed four models with BERT as a backbone for the few-shot NER task. Compared to other datasets, BERT achieved promising results, though it struggled on FEW-NERD, which is suggested to be due to the larger number of types.

General-purpose NER algorithms have long been studied. However, it may not achieve similar results when applied to the cybersecurity domain (66). Though deep learning models can leverage transfer learning, it cannot be assured that they sustain it across different domains. This motivates the creation of domain-specific corpora for named entity recognition.

MalwareTextDB (63) introduced a dataset built from annotated malware reports using the MAEC vocabulary. The reports were taken from APTnotes, a publicly available repository that consisted of 39 reports and 6,819 sentences on Advanced Persistent Threat groups. Though the repository contained 384 reports as of writing, the authors used 39 of them from the year of 2014.

DNRTI (67) was a public dataset with over 300 reports annotated in IOB notation for 13 threat intelligence-related entities. The dataset was employed to analyze four SOTA models (68). The paper tested Transformer models, such as BERT and XLNet, with the dataset, showing they perform better than the Bi-LSTM model for NER on the dataset. For this task, DNRTI has been adapted to 11 entities.

The need for a CTI corpus for training AI-based cybersecurity systems was also discussed in Hanks *et al.* (69). The authors created a publicly available dataset for NER for arbitrary entities on their ontology. Six human annotators classified 1,339 sentences. The texts were obtained from scraping popular vendor blogs. The total amount of unlabeled text is 380 articles totaling 25,000 sentences.

APTNER (18) was a dataset for NER tasks on the CTI domain. It complied with STIX 2.1 to facilitate downstream research tasks in CTI. It contained 10,984 sentences, which resulted in 39,565 entities. The authors proposed 21 entities, 7 SDO and 13 SCO, plus an "other" class, labeled in BIOES format. The workforce was composed of 30 undergraduate and 4 graduate students for labeling and 2 graduate students for verification. The dataset was evaluated by comparing six models, DNNs and BERT, in the NER task, resulting in the best performance of BERT+BiLSTM+CRF.

In Siracusano *et al.* (22), a dataset comprised of 204 publicly available reports of 62 different sources was presented. It also provided aCTIon, a solution based on generative LLM to retrieve STIX objects. It was evaluated on the `gpt-3.5-turbo` model focused on Malware, Threat Actor, Target, and Attack

Pattern entities. The collection had roughly 36,100 entities and 13,600 relations. The study noted the inherent cost of performing manual annotations by expert CTI analysts. The lack of a benchmark for many CTI tasks was highlighted as well. Many solutions to entity extraction were proposed, but addressed different problems. Though the paper addressed the necessity of open large datasets, the dataset itself was not found, and the link to it was hidden for anonymity, as stated.

Deka *et al.* (70) introduced AttackER, a dataset for cyber-attack attribution. This dataset was labeled for NER using STIX version 2.1. It consisted of a subset of 14 entities from 18 SDO and 4 entities adapted to allow a more detailed description. A total of 2,640 sentences were annotated from the initial 217 documents. Two annotators performed the labeling task with 83.8% of agreement. AttackER was used on the NER task using BERT and generative models. Three of the BERT models were finetuned previously in the cybersecurity domain. SecureBERT achieved the best result on the F1-score. For generative models, the author raised considerations on the ground truth and misalignment. These models showed promising results on zero-shot learning.

Table 2.4: Dataset Overview

Dataset	**Data Sources**	**Classes**	**Entities**	**Relations**	**Publicly Available**
MalwareTextDB (63) 2017	39 reports 2,080 sentences	4 + 444	10,983 + 7,102	8,705	Yes
DNRTI (67) 2020	300+ reports 6,570 sentences	27	36,412	–	Yes
Hanks et al. (69) 2022	380 reports 1,339 sentences	29	801	–	Partial
APTNER (18) 2022	10,984 sentences	21 STIX	39,565	–	Yes
Siracusano et al. (22) 2023	204 reports	9 STIX	36,100	13,600	No
AttackER (70) 2024	217 reports 2,640 sentences	14 STIX + 4	7,026	–	Yes
YACSDB$_{NER}$ 2025	422 reports 29,878 sentences	8 SDO	15,140	–	Yes

3 METHODOLOGY

Using LLM models is the main method explored in recent studies for NER in the cybersecurity domain, as seen in section 2.4. The SOTA methods mentioned involved LLM at some point in their pipeline. When considering BERT models, pre-training these models requires large amounts of text, which poses a limitation for this approach. However, BERT can benefit from DAPT and further downstream, which involves less but labeled specialized texts.

To explore this latter option, we addressed the lack of publicly available datasets, as observed in our review (9, 20, 12, 21, 22), by generating a dataset suitable for training BERT models. This chapter describes the process for generating a dataset for NER in the cybersecurity domain. First, design decisions are detailed regarding the source input and the output format. Subsequently, the dataset generation pipeline is elaborated, beginning with the preprocessing step. The following subsection presents a preliminary analysis of the data and discusses further design decisions. In sequence, the classification process is explained, alongside the manual review of the data. Finally, the generated dataset is described.

3.1 DATASET GENERATION

When creating a dataset for NER, design decisions must be made to ensure its effectiveness for training purposes. It should be specified what the primary objective is, which source of data will be used, which ontology or standard will be used for tagging, and which format will be employed. This approach ensures that the dataset will be well-suited for training NER models that can effectively support the classification task. After clearly defining the problem, we applied a systematic approach for generating the dataset consisting of five steps: collection, data preparation, evaluation, annotation, and review. This process is depicted in Figure 3.1.

Figure 3.1: Dataset generation process.

We observed different goals in NER studies of section 2.4. STIXnet (21) extracted IoC and leveraged rcATT (64) for recognizing TTPs. Siracusano *et al.* (22) retrieved STIX objects that were mainly threat-related entities and one for TTP. KnowCTI (33) focused on extracting a knowledge graph with their proposed ontology containing threat-related entities, IoC, and TTP, as well as the relations among them.

There is a complication when comparing the results of the studies because of a deficiency of benchmarks. Different classification problems need different datasets, for instance, for sequence tagging, NER, RE, and knowledge graph generation tasks. Since NER is a preceding problem for the others, it was addressed.

There is an extensive landscape of CTI standards, as seen in section 2.2. This is observed on the available datasets, where many schemas and ontologies were proposed. So choosing a common language is an important decision for the project design. We selected the STIX language due to its consistent reference in studies.

The first scope of the dataset was labeling SDO as the coarse-grained classes. The token tagging format was used because converting to word or span tagging from it is straightforward, which facilitates broader usage of the dataset. Analyzing the classes present in STIX SDO, Attack Pattern is related to TTP and is usually described in a multi-word span; moreover, it may overlap with other classes. The single-class problem was initially addressed, which justifies investigating the Attack Pattern in future work.

When analyzing cybersecurity reports, they frequently do not encompass all objects. This study focuses on retrieving information related to the threat. The following SDOs were excluded. Course of Action, which may be present on reports related to vulnerabilities, but is out of scope. Grouping works as a meta-object before the finished intelligence product; therefore, before the production of the report. Incident is a stub object not defined on STIX version 2.1. Infrastructure would be listed as an Indicator. Intrusion Set expects more than one consistent event, while there are reports describing it, there is no objective criterion to differentiate it from campaigns, which will be considered in the study. Malware Analysis expresses one specific analysis of a sample that will be described in the intelligence product and was excluded to avoid redundancy. Note is a subjective annotation; therefore, it is out of scope. Observed Data is more related to raw data; on reports, it is expected to be Indicators. Opinion falls in the same category as Note. The texts analyzed are of type Report, meaning all the extracted objects would be related to a Report object.

After excluding objects that aren't relevant to the data, the set of entities sought to be extracted is: **Attack-Pattern**, **Campaign**, **Identity**, **Indicator**, **Location**, **Malware**, **Threat-Actor**, **Tool**, and **Vulnerability**.

Another key design decision is the dataset format. There are many common tagging formats for representing multi-word named entities (71). One way of representing is assigning each named entity token a tag composed of a prefix and a class, while the remaining tokens of the span are assigned a special class. The purpose of this scheme is to add relative position information for each token. A common set of prefixes used is described in Table 3.1.

Table 3.1: Common annotation prefixes

Prefix	**Meaning**	**Description**
`B`	Beginning	Represents the first token an entity.
`I`	Inside	Represents any token within a multi-token entity besides the first.
`E` or `L`	End or Last	When present, represents the last token of a multi-token entity.
`O`	Outside	Represents a token that is not part of any named entity.
`S` or `U`	Single token or Unit	When present, represents a single token entity.

The tagging schemas are different combinations of the prefix in Table 3.1, and using each one yields different performance (71). There is no clear evidence of a better representation for NER in an English cybersecurity corpus; therefore, the IOB notation was arbitrarily selected. In this notation, single token entities along with the first token of multi-token entities are assigned the "B-" prefix, any token besides these in multi-token entities is assigned "I-", and the remaining non-entity tokens are assigned "O".

There are many sources of unstructured cybersecurity texts. The goal is to build a comprehensive enough dataset. Instead of relying on the selection and collection of sources, we selected a well-known single collection of sources. VX-underground is a website about malware and cybersecurity containing a series of reports on APTs split by year. Sources may be security vendors' blogs, CERT incident reports, X (former Twitter) posts, private researchers' reports, and others.

The pipeline that implements the process in Figure 3.1 for building YACSDB has four steps: collection and preprocessing, processing and analysis, classification process, and manual review. A visual depiction of the process is illustrated in Figure 3.2.

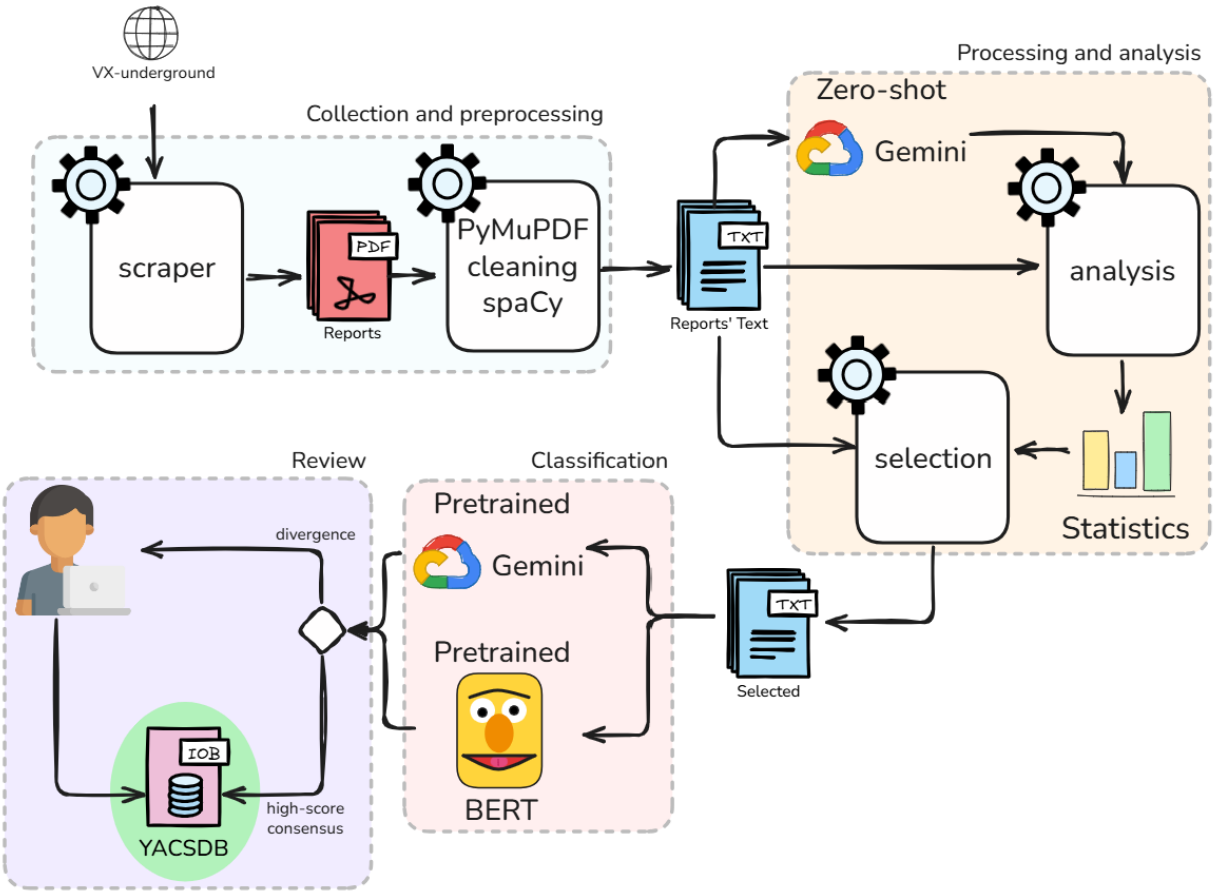


Figure 3.2: YACSDB$_{NER}$ generation pipeline.

3.2 COLLECTION AND PREPROCESSING

The first step of the pipeline consisted of scraping the reports from the VX-underground repository, extracting the text from files, cleaning data, and selecting sentences, as seen in Figure 3.3.

There were 2,164 reports listed from 2010 to 2023, which were scraped using Python, Selenium, and BeautifulSoup. Due to duplication, broken links, or empty files, the total number of files retrieved was 2,068 files in PDF format.

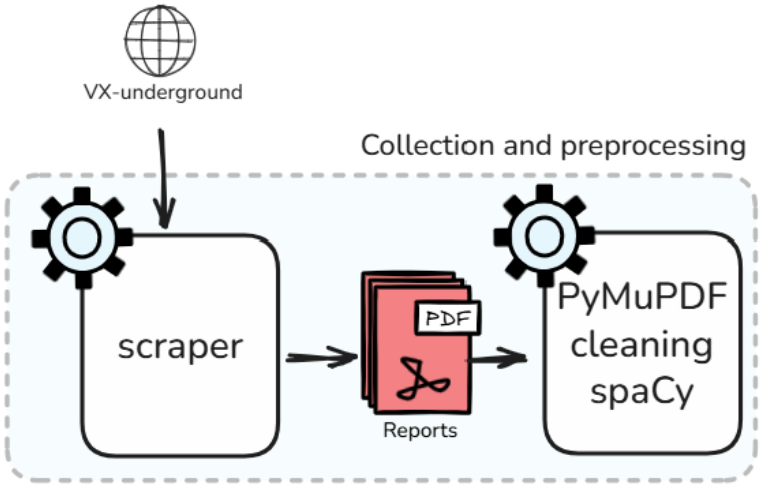


Figure 3.3: Reports collection and data preparation

After acquiring the reports, their texts were retrieved employing the module *fitz* from PyMuPDF[1]. The selected output for text extraction was natural reading sort.

Some reports contained Unicode characters that would interfere with BERT embedding, such as Arabic or Cyrillic script, and some symbols. The paragraphs containing these characters were stripped to avoid inputting partial sentences. Each paragraph was split into sentences using spaCy's[2] English general-purpose large model. Only sentences with verbs were considered for this step. We retrieved 314,323 unprocessed sentences, of which 263,066 remained valid for analysis.

3.3 PROCESSING AND ANALYSIS

In the second stage of the pipeline, presented in Figure 3.4, the objective was to select a subset of the valid reports for further classification. The reports must be representative of real-world reports to enhance the modeling of the problem. Another requirement is to have sufficient instances of each class, as data scarcity negatively impacts the model training (72). These requirements pose a challenge: how to know if the analyzed report attends to the criteria?

The approach chosen for this initial problem was to handle the entity identification task, a binary classification of tokens. Our premise was that a high entity count in a report has a higher likelihood of having more instances of each class. Therefore, we want to model real-world entity density by observing the found entity density. A naïve approach is proposed, employing a large generative model as a discriminator.

Inspired by AttackER (70) results using generative models in NER tasks, we exploited a zero-shot learning approach on the Gemini Flash 1.5 model to assess the quality of the reports regarding class coverage. Zero-shot classification in generative models is a method by which the instruction is crafted with

[1] <https://pymupdftest.readthedocs.io/en/stable/intro.html>
[2] <https://spacy.io/>

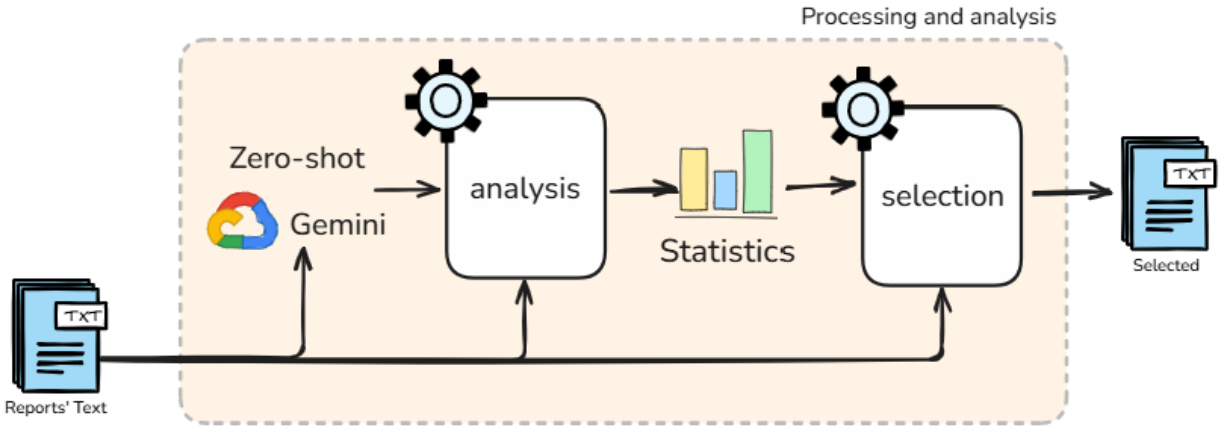


Figure 3.4: Evaluation step.

explicit information about the classes and task to be performed. It is a type of prompt engineering to leverage the model's capability in text tasks.

The strategy consisted of role-playing a cybersecurity expert to direct the model to a cybersecurity context. Besides, the output format was delimited to allow easier parsing. To mitigate hallucinations, the classes were restricted to a fixed set, a specific output was determined for cases where no entities were found, and the order of entities was enforced. The instruction utilized in the zero-shot classification is illustrated in Listing 3.1.

Listing 3.1: Instruction for zero-shot classification

```
1  You are a cybersecurity expert. You are going to extract STIX Domain Objects from
       cybersecurity reports.
2  Only output a the entity found on the text followed by its STIX SDO classification
        on the format entity:classification.
3  Output it as a list separated by a new line.
4  Don't filter duplicates, output all the entities in the same order they appear in
       the text.
5  The text may not have any entity, in which case you will output only "None".
6  The classes you will use for classification are the following:
7  intrusion-set
8  location
9  attack-pattern
10 identity
11 tool
12 malware
13 campaign
14 indicator
15 vulnerability
16 threat-actor
```

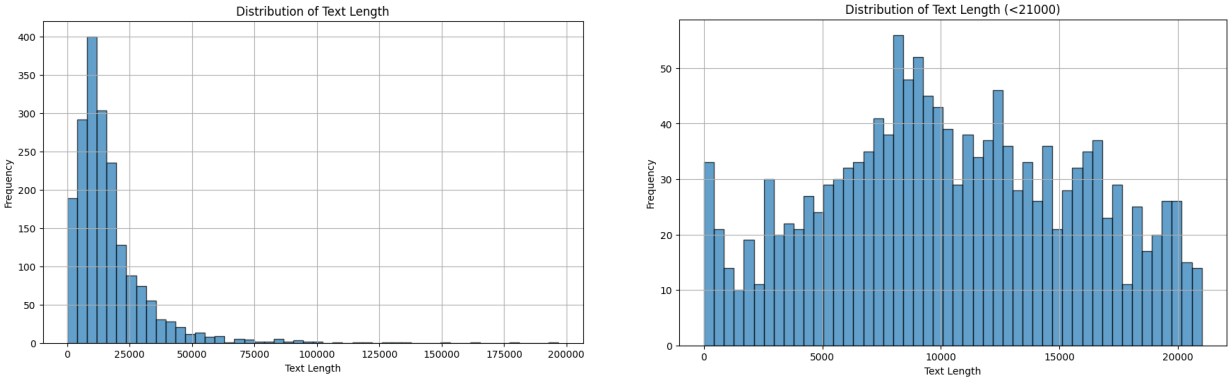
Gemini models show good performance in finding entities, such as in the MMLU task (57). For this preliminary metric, the identification of the entity suffices, even if the model misclassifies them. The

objective is to evaluate the texts based on the number of STIX entities found as a previous metric, based on our premise of a higher likelihood of label coverage. Each valid sentence from a report was used as input. Due to the nature of the content being related to threat reports, safety thresholds configuration of dangerous content (HARM_CATEGORY_DANGEROUS_CONTENT) and as well as the configuration regarding hate speech (HARM_CATEGORY_HATE_SPEECH) were set not to block (BLOCK_NONE). To handle hallucination, each entity was cross-checked within the text, and only found entities were considered. The total number of reports that successfully extracted entities was 1,924. The total number of entities retrieved from the selected classes after sanitization was 233,899.

3.3.1 Reports descriptive statistics and analysis

A detailed characterization of the reports with found entities is provided. We present key descriptive statistics that helped in the decision-making of the dataset design. It was set that the goal of the initial dataset was to understand the landscape of cybersecurity reports. To achieve this objective, the data was curated to minimize the presence of outliers and enable the exploration of the inherent properties of the most representative reports.

The text length ranged from 10 characters to $196,840$ in the longest report. While there was a large difference between the shortest and the longest, the text length distribution had a great positive skewness, with a mean length of $16,921.2$ and a median of $12,706$. The 90^{th} percentile is close to $33,000$, which is characterized by the long tail on Figure 3.5a. Figure 3.5b shows a close to zero skewness when it is taken into consideration only reports smaller than $21,000$ characters, which is approximately the 75^{th} percentile.



(a) Full set

(b) Under 21,000 characters

Figure 3.5: Distribution of reports' length (character count).

The number of reports available per year is displayed in Table 3.2. It can be noted that the years until 2014 had a low number of examples, while 2022 had a spike in reports. The mean count of reports from 2015 to 2023 is approximately 200. It is not known the reason for the lower count of reports in the first years, for instance, whether it was a problem in collection.

From the observation of the number of valid entities in Gemini's output in Figure 3.6, it can be noted a positive correlation between the number of entities and the length of the reports. Additionally, the relation-

Table 3.2: Initial number of reports per year

Year	Reports	Year	Reports	Year	Reports
2010	5	**2015**	136	**2020**	184
2011	14	**2016**	140	**2021**	162
2012	24	**2017**	117	**2022**	437
2013	45	**2018**	162	**2023**	282
2014	31	**2019**	185		

ship indicates a linear trend with high variability. We delve deeper into the analysis of these characteristics to address the representativeness of the reports.

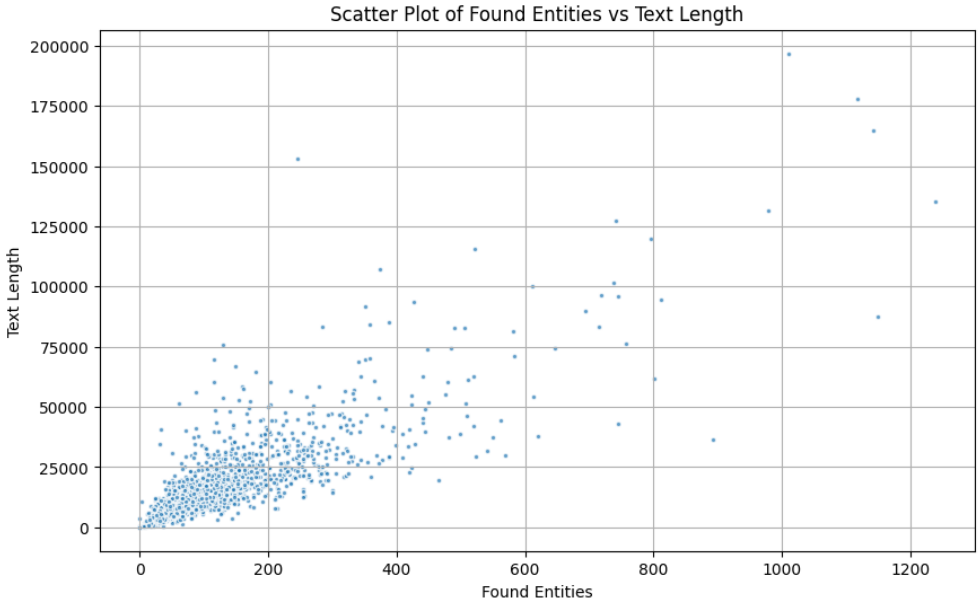


Figure 3.6: Ratio of Entities per Text Length (EPTL) versus text length.

A preliminary metric was proposed to enable assessing the expected quality of the reports regarding representativeness and label coverage. The extracted entities per 1,000 characters (EPTL) is a ratio of entity density with the assumption of positive correlation with the label coverage. The analysis of this metric allows the comparison between reports, regardless of their size.

As there are more instances of smaller texts, a greater variability for these texts can also be observed in Figure 3.7. Yet, there is an observed trend for the entity density to be concentrated in the range of 0 to 25. A yearly breakdown, presented in Figure 3.8, indicates a growing saturation in the range of 5 to 10.

An initial selection of the reports was done to allow a manageable amount for the first version of the dataset. Hence, we aim to optimize processing efficiency and maximize report variety. Longer reports had a rather uniform entity density. As the initial goal is not a document-level task, reports smaller than 21,000 characters were selected. The number of instances in the first years was substantially lower than in more recent years. It is not known the reason for this, for instance, if it was a problem in collection, which brings a question of whether those were representative. Taking into consideration that older data might bring an outdated bias, we selected only reports after 2014. A close examination of the entity density of some outliers showed a set of documents that were primarily composed of IoC listings. It was observed

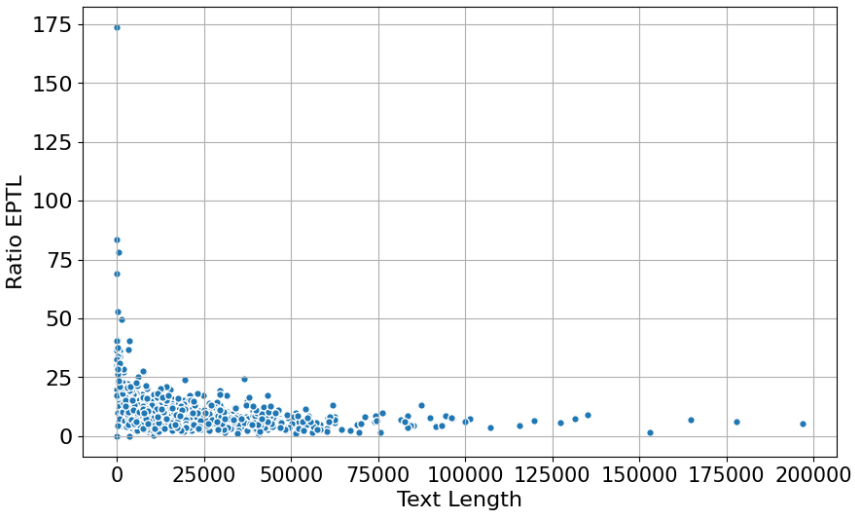


Figure 3.7: Ratio of Entities per Text Length (EPTL) versus text length.

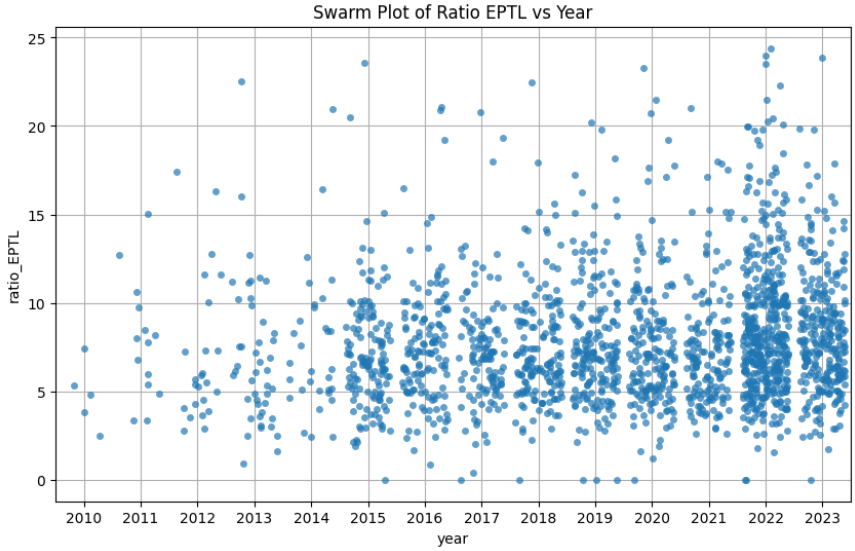


Figure 3.8: EPTL distribution by year.

that selecting reports with an EPTL value ranging from 5 to 20 covered enough variety. After applying all criteria, the initial version of the dataset comprises 1,127 candidate reports, as detailed in Table 3.3. This is a stratified sampling, considering the proportion of instances per year.

Table 3.3: Number of reports per year

	2015	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**	**2022**	**2023**
Reports	73	76	77	106	120	95	85	300	195

3.4 CLASSIFICATION PROCESS

In the classification step, we propose an LLM-assisted methodology for building the dataset for NER. The classification task was performed with the assistance of pretrained LLM models within the dataset, as depicted in Figure 3.9. We leveraged Gemini and BERT models by gradually enhancing their adaptation to the cybersecurity domain. The primary objective was to assist cybersecurity experts in labeling with limited resources.

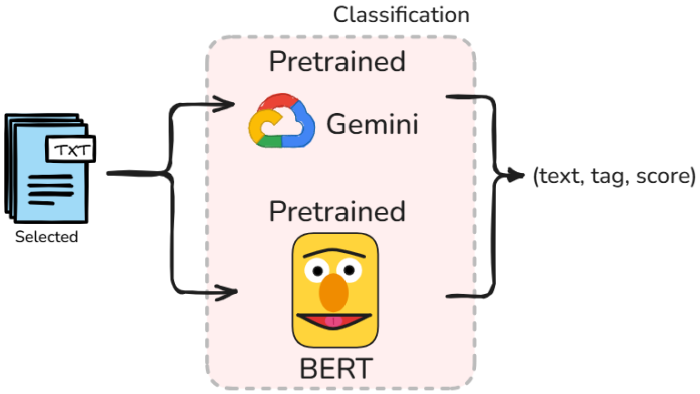


Figure 3.9: Assisted classification step.

Fine-tuning LLM is computationally expensive; thus, the Parameter-Efficient Fine-Tuning (PEFT) method is preferred to downstream these models (73). Bearing this in mind, Gemini models provide support to fine-tune them and, while it is not clearly stated in their documentation, the method used for this is likely Low-Rank Adaptation (LoRA)[3] (74).

To fine-tune Gemini, a training set of input-output pairs should be submitted for the specific task. For a zero-shot prompt approach, similar to the Section 3.3, the input examples should mirror the instructions that will be provided for inference, enabling the model to internalize these patterns. Google recommends providing a minimum of 100 examples for fine-tuning in a classification task and up to 500 for a summarization task[4].

Since there is a scarcity of labeled data, we adopted a pseudolabeling strategy. Pseudolabeling is a semi-supervised learning method where a model trained on a small label set is used to generate labels for an unlabeled dataset. Afterwards, labels with high confidence are used for further training.

The pseudolabeling can be described as the following iterative process. At each iteration n, the model M_n is initialized from the final last state of the preceding model M_{n-1}. Next, M_n is fine-tuned on dataset D_n. This dataset D_n comprises pseudolabels that were generated by M_{n-1} on an unlabeled sample set $\mathcal{X}_n$. Let $M_{n,\theta}$ denote the specific instance of the Gemini or BERT model at iteration n with parameters θ, and let $\mathcal{L}$ represent the loss function applied to M_n on D_n. The fine-tuning process can be expressed as shown in Equation 3.1.

[3] <https://medium.com/google-cloud/fine-tuning-gemini-best-practices-for-data-hyperparameters-and-evaluation-65f7c7b6b15f>
[4]<https://ai.google.dev/gemini-api/docs/model-tuning#size-recommendation>

$$M_n = \arg\min_{\theta_n} \sum_{i=1}^{n} \mathcal{L}(M_{i-1,\theta_{i-1}}, D_{i-1}) \tag{3.1}$$

The central issue of applying pseudolabeling to fine-tune Gemini is that it does not output the probability of each token; hence, it is not possible to assess the confidence of the classification. Only the average log probability, which is the average log probability of all tokens, is accessible. To overcome this matter, a BERT model was also fine-tuned similarly as follows.

The initial base model for training M_0 was `gemini-1.5-flash-001-tuning`. This model was trained in our initial dataset D_0, derived from STIXnet's evaluation dataset (21), which contained 52 APT reports with 1,407 entities. The dataset did not fully comply with STIX SDO; thus, we made minor adaptations to this dataset so it strictly aligned with SDO. Since this dataset was going to be used in BERT's training as well, the texts were divided into spans of at most 128 tokens using the BERT tokenizer, which resulted in 152 tagged spans.

Each entry was preprocessed before feeding it to the model. The source span was concatenated to an instruction template, shown in Listing 3.2, to be used as training input. The training output is the list of named entities and their classes, one per line.

Listing 3.2: Instruction template for training

```
1 Task: You are a cybersecurity expert. Extract STIX 2.1 Domain Objects.
      Your task is Named Entity Recognition.
2     Format: <label>:<entity>
3     One entity per line.
4     Labels: location, identity, tool, malware, campaign, indicator,
          vulnerability, threat-actor
5     Text:
```

We executed three successive rounds of fine-tuning on the Gemini model. At each round, a small unlabeled sample was selected for inference. By doing this, the performance of model M_n gradually enhanced, and only a small number of errors needed to be fixed. Therefore, it minimized any error that might be propagated. This iterative process built an initial ground truth with enough number of instances as recommended by Google. The resulting dataset contained 416 spans. Hyperparameters used are in Table 3.4 and all models had the temperature set to 0.3. We started with a low learning rate for model M_1 to avoid overfitting, as the initial dataset was small, and the texts showed some similarity.

Table 3.4: Training parameters used in Gemini fine-tuning

Parameter	M_1	M_2	M_3
Epoch Count	5	10	10
Batch Size	4	4	4
Learning Rate	0.001	0.0005	0.0005

Along with Gemini, a set of BERT models was trained in the same fashion over the same datasets. BERT resulting score for each label was used as prediction confidence. To assess the results of both Gemini and BERT models, their outputs were ingested in LabelStudio[5], an open-source data labeling tool.

The complete iteration process is depicted in Figure 3.10. The last Gemini model M_3 and the fine-tuned BERT model were applied to a subset of 380 reports from the candidates. The resulting dataset was then evaluated.

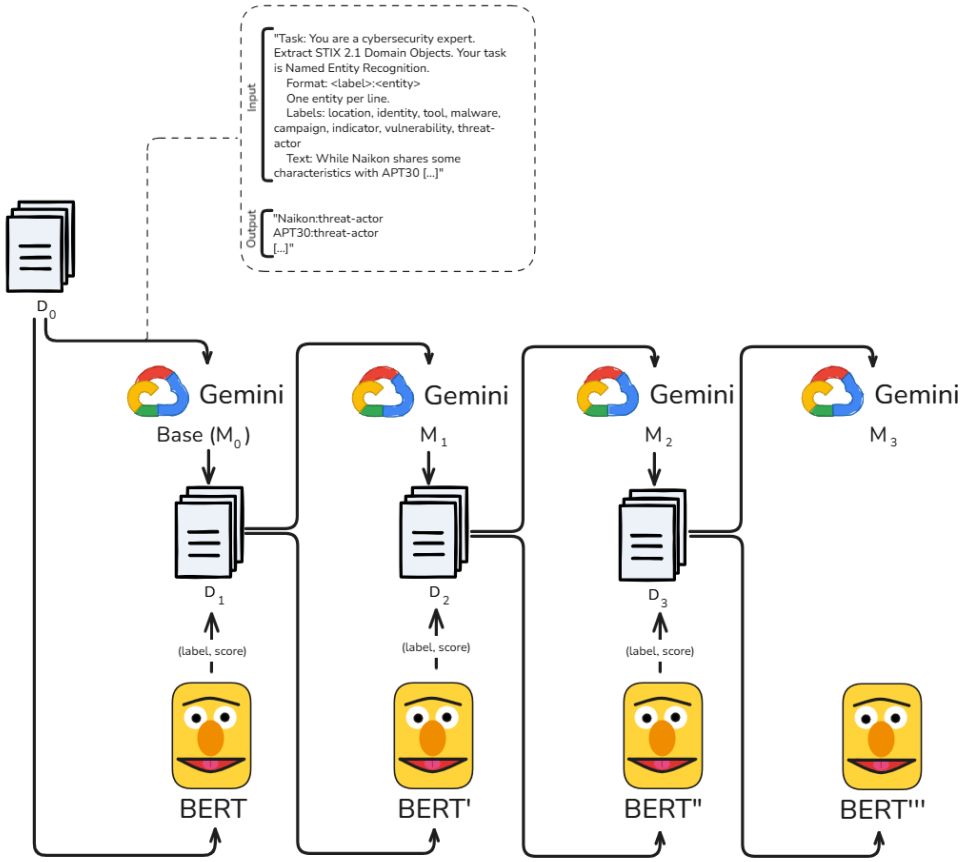


Figure 3.10: Iterative training of the models.

3.5 MANUAL REVIEW

The last stage of the pipeline is the manual review. This step mitigates the risk of propagating errors. Additionally, it inserts the critical analysis of the annotator with experience in the field. In a complex domain such as cybersecurity, it allows for verifying the outputs of text with ambiguous semantics.

LabelStudio enabled direct comparison of the models' results. The inference of both models was input to the tool. Along with it, entities found across multiple reports had their labels checked to assert the correctness. When the models diverged in their labeling, the annotator selected the most appropriate label.

[5]<https://labelstud.io/>

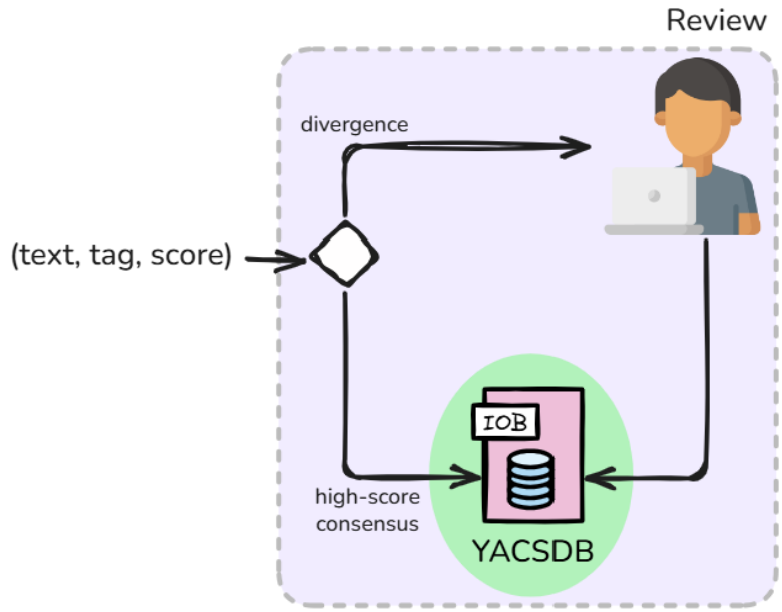


Figure 3.11: Review step.

The annotator had 10 years of experience in cybersecurity. Spans without entities are removed to favor higher label density. The F1-score for the Gemini M_3 model alone yielded an F1-score of 0.904.

3.6 YACSDB$_{NER}$

YACSDB$_{NER}$ is composed of two parts. The first small set is the reviewed STIXnet's evaluation dataset used as the initial training dataset. Since it was manually reviewed to comply with SDO and also used in training, this dataset was incorporated into the corpus. The second, larger part is the dataset built from the manual review of the proposed labeling methodology.

The key characteristics of the dataset are summarized in Table 3.5. The sentence count was obtained using the spaCy model, as detailed in Section 3.2. Spans were limited to a maximum of 128 tokens using the BERT tokenizer. The total number of classes considered all labels, with both 'B-' and 'I-' prefixes and the 'O' class, adding up to 17 classes, whereas the total entities count considered the 8 SDO entities, taking into account Subsection 2.2.4 and Section 3.1 regarding Attack Pattern. Figure 3.12 presents an example of an annotated sentence in the dataset.

Table 3.5: YACSDB$_{NER}$ description

Sentences	24,878
Spans	8,169
Total Entities	15,140
Number of Classes	17
Source Reports	422

```
EclecticIQ B-identity
researchers O
followed O
the O
latest O
activities O
carried O
out O
by O
the O
Dark B-threat-actor
Pink I-threat-actor
APT I-threat-actor
group O
and O
identified O
how O
the O
group O
further O
honed O
its O
technical O
skills O
to O
bypass O
security O
controls O
, O
scale O
TTPs O
for O
, O
blend O
in O
with O
victim O
environments O
, O
and O
hinder O
detection O
across O
all O
aspects O
of O
its O
operations O
. O
```

Figure 3.12: An annotated sentence.

4 EXPERIMENTS

The experiments that were carried out are presented in this chapter. The first experiment sets the baseline for the remainder of the study. By observing the results of fine-tuning BERT models in a cybersecurity dataset, the main goal of the study, we established the design of the following steps. This analysis supported the strategy for the experiment with Gemini models. Based on the available information at the time, the best-performing Gemini model was selected from different training strategies.

The main experiment is to employ the YACSDB$_{NER}$ dataset on fine-tuning of different models. Detailed results enabled the comparison among them as well as to assert possible limitations of the dataset. Each model is scrutinized to understand the outcomes.

4.1 BERT FINE-TUNE EXPERIMENT

Before engaging the classification task with Gemini, an initial assessment was performed with BERT models applied to STIXnet's evaluation dataset. The dataset only had 152 spans, which is substantially low when considering the studies in Section 2.4. Such a small dataset most probably would lead the model to overfit. Nevertheless, this small assessment could provide insights for a first design.

BERT is the baseline model for many studies. For this study, we also selected the BERT base cased model as a baseline for the initial dataset. Along with it, we also selected SecBERT and SecureBERT to allow comparison. The models were loaded from the Hugging Face *Transformers* package.

Two different approaches to train BERT were taken into consideration. In the first instance, the training loop code was developed to ensure a basic start. The process involved running the training function on all batches for the chosen number of epochs. Afterwards, the model was put in training mode so the weights could be adjusted, the accuracy of inferences was computed for active labels, backpropagation was performed, and optimization was applied. The second method consisted of employing Hugging Face's Trainer class. This class implements the training using the passed parameters.

The results for this assessment are illustrated in Figure 4.1. The models were trained for five epochs with a learning rate of $1\mathrm{e}{-5}$ and a train batch size of 4. The model with the lower train loss was a BERT base cased model trained with Trainer. Therefore, this setup was later explored in a classification task.

4.2 GEMINI EXPERIMENT

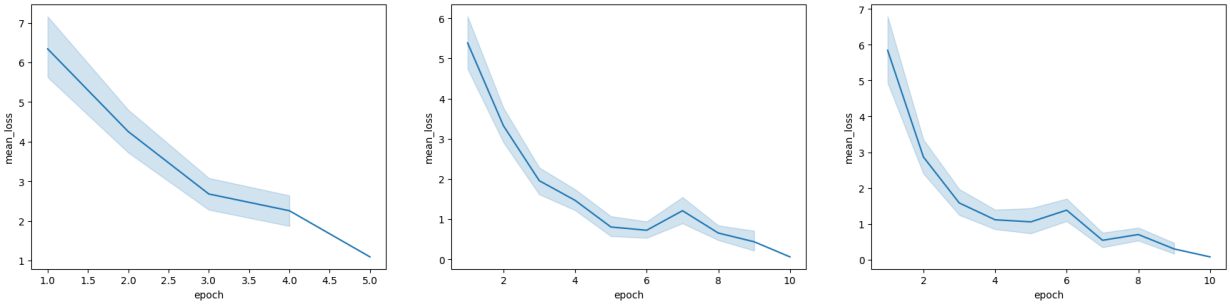
Before proceeding with the classification task of Section 3.4, three different approaches were tested. Though the iteratively trained model M_3 was expected to have good performance, a cautious investigation was deemed necessary.

(a) Custom implementation

(b) Hugging Face's Trainer implementation

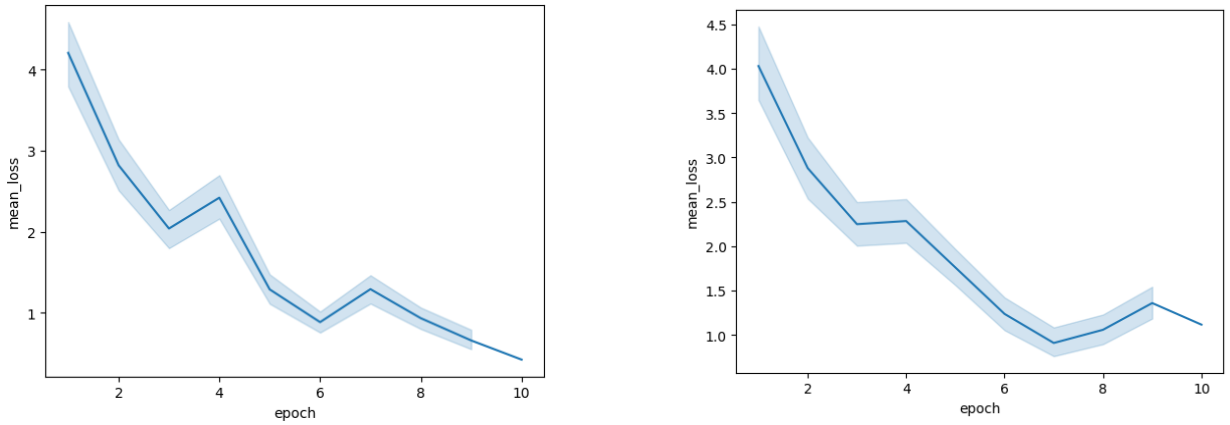
Figure 4.1: Training loss curve for BERT fine-tuning in STIXnet's dataset.

Once we had a large enough dataset for training Gemini, two additional models were trained on this dataset. The first model, $M_{all,4}$, was trained on the entire dataset in a single job, utilizing the same hyperparameters as model M_3, detailed in Table 3.4. Similarly, the second model, $M_{all,8}$, was also trained in the entire dataset in a single phase, employing the same hyperparameters as M_3, except the batch size, which was set to 8.



(a) M_1 (b) M_2 (c) M_3

Figure 4.2: Mean loss for models from Table 3.4.



(a) $M_{all,4}$ (b) $M_{all,8}$

Figure 4.3: Mean loss for models training in the whole dataset.

Figure 4.2 presents the mean progression for the models trained through an iterative process. The iterative approach aimed to gradually enhance performance while mitigating overfitting on the small training dataset. Conversely, Figure 4.3 depicts the mean loss curve for the models trained on the complete training dataset. The last value of the mean loss for each model is shown in Table 4.1.

Table 4.1: Mean Loss for Gemini Models

Model	Mean Loss
M_3	**0.075**
$M_{all,4}$	0.423
$M_{all,8}$	1.116

Model M_3 seems to benefit from its iterative approach and converges to a lower loss. Additionally, there is a slight difference in train stability among the models. Since lower loss generally indicates a more accurate prediction, and training stability points to more consistent learning, model M_3 was selected to be employed in the classification task of Section 3.4.

4.3 YACSDB$_{NER}$ EXPERIMENT

This experiment was conducted to assess the quality of the dataset for fine-tuning and benchmarking models. To assess the utility of the dataset for NER, we fine-tuned seven BERT-like models on this task. The laboratory environment consisted of a Google Colab Python 3 environment with an NVIDIA T4 GPU.

We split the dataset into a 70:15:15 ratio for training, validation, and testing sets. As a baseline, we employed BERT$_{frozen}$, a BERT base cased model with frozen weights where only the classifier head was trained. This practice enables us to compare how the BERT model performs in this specific task before and after fine-tuning, isolating the contribution of the training. Additionally, we evaluated a standard BERT base cased model without any self-supervised DAPT. These models were compared against CyBERT (12), CySecBERT (19), and SecBERT, which are three DAPT BERT versions, as well as against SecRoBERTa and SecureBERT (17), which are two DAPT RoBERTa versions. All these models are domain-adapted to cybersecurity datasets.

For NER, the simplest arrangement is to apply a simple classifier head $Linear(768 \rightarrow 17)$ on top of each model, following the original BERT (7) approach. These models were fed with identical training, validation, and testing datasets. To run the fine-tuning, we employed Hugging Face's Trainer, as in Section 4.1, passing the same hyperparameters: 5 epochs, a training batch size of 8, a validation batch size of 2, and a learning rate of $1e-05$. All other parameters were set to their default values.

4.4 RESULTS AND DISCUSSION

The models' computed metrics are presented in Table 4.2. These are the outcomes of training with the YACSDB$_{NER}$ dataset. BERT$_{frozen}$ results highlight the complexity of the cybersecurity domain and the need for adaptation specifically to this domain. Furthermore, when comparing its performance to other

Table 4.2: BERT models results

	Precision	Recall	F1-score
BERT$_{frozen}$	0.136	0.000*	0.001
BERT	**0.767**	**0.806**	**0.786**
CyBERT (12)	0.578	0.629	0.603
CySecBERT (19)	0.567	0.630	0.597
SecBERT	0.552	0.519	0.535
SecRoBERTa	0.572	0.528	0.549
SecureBERT (17)	0.750	0.791	0.770

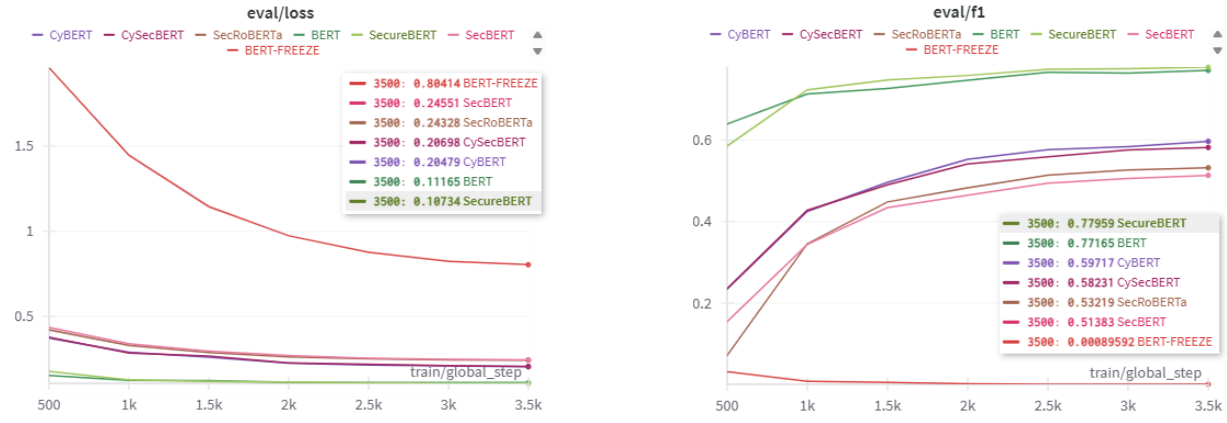
*value is small, but not zero

models, it indicates that this dataset is suitable for NER. Since each was computed from the same starting point, this dataset can serve as a benchmark for evaluating them.

The general-purpose BERT outperformed all DAPT models in all metrics. Its results were closely followed by the SecureBERT model, which is a RoBERTa model. The performance of the remaining DAPT models was notably suboptimal.

Both SecBERT and SecRoBERTa were pretrained on the same dataset. These models have close performance, with a slight advantage to the RoBERTa model. Even so, they have the lowest F1-score in Table 4.2. CyBERT and CySecBERT are adapted from BERT models, yet all these four models achieve similar precision scores.

To help understand the results, further metrics are analyzed. The performances of BERT and SecureBERT are much alike during evaluation, as depicted in Figure 4.4. SecureBERT has even a negligibly lower loss and a higher F1-score than BERT in the evaluation step, indicating the models have comparable results.



(a) Loss curves

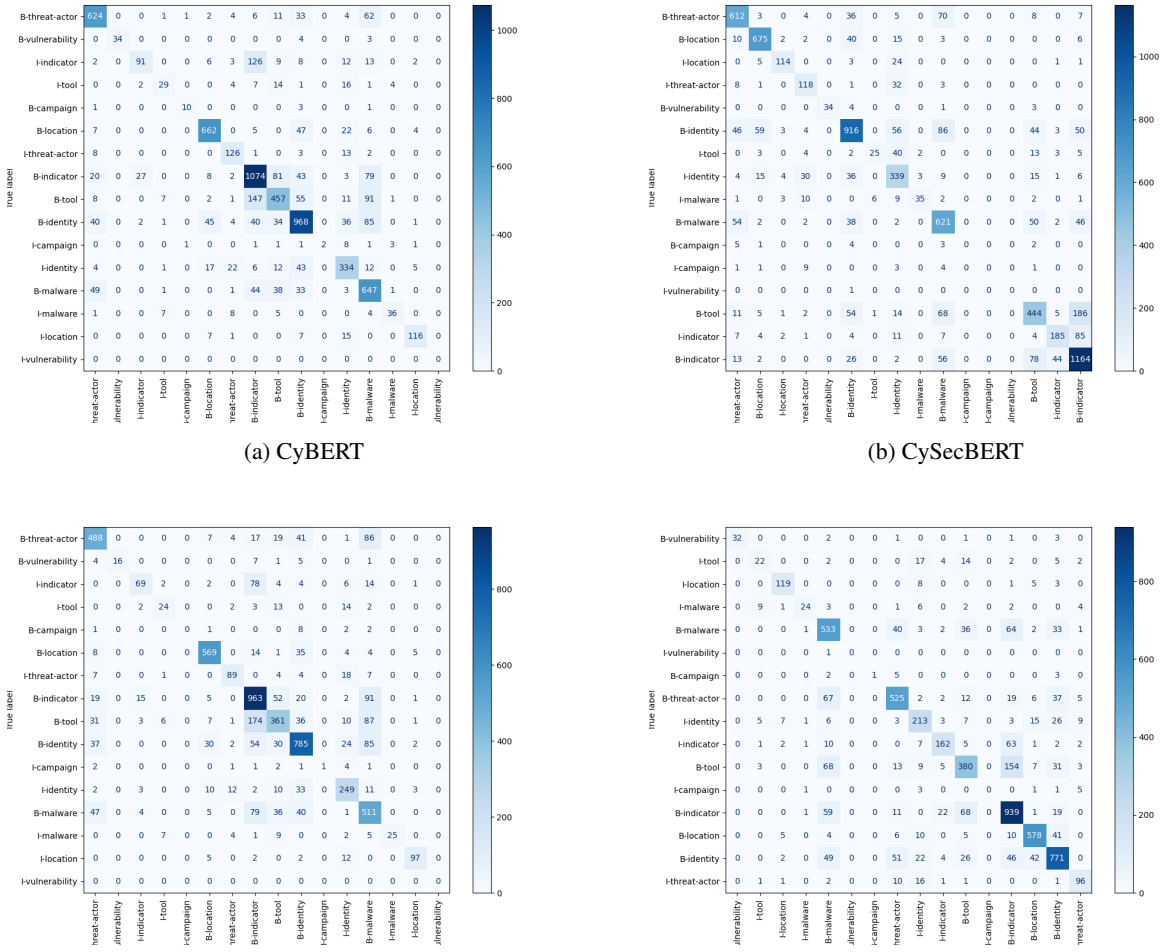
(b) F1-score curves

Figure 4.4: Evaluation scores.

A confusion matrix visually represents the performance of classification models. In multi-class classification, it offers insights into classification errors and potential areas for model improvement. Predicted labels are aligned with the columns, while the true, expected labels correspond to the rows. Figure 4.5 illustrates the confusion matrices generated by the suboptimal models, whereas Figure 4.6 presents the

confusion matrix for SecureBERT. It is important to note that the order of the labels is not consistent across these figures.

At first, an imbalance is observed among classes. Underrepresented classes are harder to model, which leads to unpredictable behavior. For instance, the class "B-campaign" is most classified as "B-identity" by SecBERT, "B-threat-actor" by SecRoBERTa, and "B-malware" by SecureBERT, while CyBERT handles it the best. SecureBERT shows some confusion between "B-indicator" and "B-malware". The models consistently have trouble with the "B-malware" class, and misclassify it as "B-threat-actor", "B-tool", "B-indicator", and "B-identity". Interestingly, mistakes occur in similar amounts for suboptimal models. The "B-tool" class is most often attributed to "B-indicator", and "B-threat-actor" to "B-malware".



(a) CyBERT

(b) CySecBERT

(c) SecBERT

(d) SecRoBERTa

Figure 4.5: Confusion Matrices

Nevertheless, this comparison also underlines a significant opportunity for improvement in the model adaptation process within the cybersecurity domain.

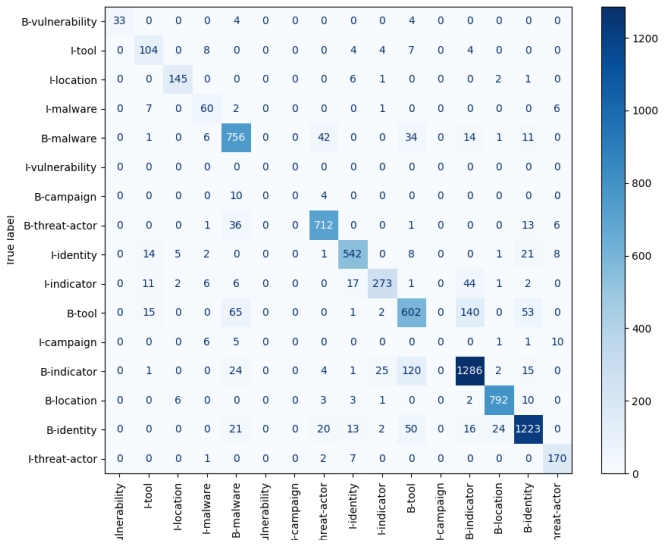


Figure 4.6: SecureBERT's Confusion Matrix

4.4.1 Discussion

While there are different goals for applying NLP in the cybersecurity domain, the analyzed research in cybersecurity related to NER shows much interest in identifying IoCs, TTPs, and CTI entities, as observed in Table 2.2. The findings are similar to the results of Rahman *et al* (13). Also, these are the main topics of datasets from Table 2.4.

RQ1. *What are the main goals to which the datasets are designed in the cybersecurity domain?*

The principal objectives of datasets in the cybersecurity domains align with tasks of IoC extraction, TTP classification, and cybersecurity-related entities extraction.

Nevertheless, the publicly available datasets are scarce, even though there are many cybersecurity texts. This is noted in most research and is frequently associated with the high need for resources for the annotation task. When the annotation is performed, usually a small team is employed. Considering this scenario, we formulate this answer.

RQ2. *Which are the common constraints in the dataset generation?*

The high need for time and human resources is a common constraint in dataset annotation. The annotation task takes a long time to complete, and most teams may not have enough specialists for the duration of the task.

To address these constraints, the labeling pipeline presented in Section 3.1 leveraged Gemini as a naïve pre-evaluation tool for the report selection, yet this selection introduced the bias of this model. Most common named entities, which are present in tasks it evaluates, such as MMLU, may not represent the performance for cybersecurity-related named entities, introducing bias towards general named entities. This comparison can be easily conducted with YACSDB$_{NER}$.

The experiments in Section 4.2 evaluated the strategies for the assistant Gemini model, indicating that an iterative fine-tuning, as run to train M_3, achieved better metrics than training on the whole dataset at once, yielding an F1-score of 0.904. The iterative training resembled curriculum learning (75) by gradually introducing representative reports in pseudolabeling. This result does not imply good generalization, considering that pseudolabeling can propagate errors within the dataset and introduce a confirmation bias. The iterative process helps mitigate those issues; moreover, cross-validation strategies, such as k-fold, can further improve this mitigation. This scenario is particularly true when it comes to the class imbalance, which was observed in Figures 4.5 and 4.6.

The assisted labeling process maximized the annotator's effort, which can be quantitatively assessed to establish a robust procedure to estimate resource needs for labeling a corpus. Approximately 25,000 sentences were annotated with a single annotator. The process, as regards resource employment, accomplishes a notable deed when compared to APTNER (18), which required 36 annotators with different skill levels to label nearly 11,000 sentences. An even larger dataset, such as Few-NERD (65), employed 70 annotators and 10 experienced experts, making this is resource-intensive task.

RQ3. *How to efficiently build the dataset?*

We propose that our LLM-assisted methodology, exploiting an iterative training, is efficient to create a labeled dataset.

However, the annotation process introduces bias that must be considered. Given the selected emphasis on higher entity density in the dataset, training strategies should consider employing measures to mitigate potential overfitting. Higher entity density does not tackle the class-imbalance problem. The scarcity of data, therefore, of instances of low-represented classes, impacts the models' performance, increasing misclassification (76). Introducing artificial instances and noise could improve the generalization capability of the models.

YACSDB$_{NER}$ presents advantages over existing public datasets for cybersecurity NER tasks. MalwareTextDB (63) focuses specifically on malware analysis; however, CTI relies on up-to-date information, and this dataset comprises older reports and does not encompass key STIX entities. DNRTI (67) improves representativeness compared to MalwareTextDB, but it still omits SDOs. Hanks et al (69) provides a public corpus without annotations, requiring additional effort for NER. APTNER (18) is a valuable resource with labels for indicators; however, sources and dates of reports are neglected, and a workforce of 36 annotators poses practical limitations, which makes an LLM-assisted annotation more accessible. Finally, AttackER (70) includes fewer sentences and annotated entities compared to YACSDB$_{NER}$.

Moreover, these datasets can advance NLP in the landscape if employed alongside. Aligning the classes in them using a common language, such as STIX, addresses the scarcity of data in the cybersecurity domain. Furthermore, NER would benefit from a comparison among the datasets to assess quality in terms os representativeness and label coverage.

The performance of fine-tuned models highlights the challenges of supervised NER in the cybersecurity domain. The number of classes may be a limitation for BERT models, as it was observed in general-purpose settings (65). Conversely, the results achieved by BERT reinforce the advantages of downstreaming in a domain. However, the observed suboptimal performance of DAPT models warrants further investigation.

This finding draws parallels with results in Zanella and Toussaint (77) in the biomedical domain, where certain models did not outperform BERT when employing linear classifiers. Notably, SecureBERT, a RoBERTa-based model, incorporates larger vocabulary modifications – representing 0.35% of total tokens, compared to approximately 0.03% for CyBERT – and benefits from pre-training on a larger corpus than both CyBERT and CySecBERT. Benchmarking with the YACSDB$_{NER}$ dataset allowed a direct comparison of models, suggesting that SecureBERT's adaptation strategy is better suited for this specific task. The remainder DAPT models may have underperformed due to a mismatch between the linguistic style of their pretraining corpus and that of the YACSDB$_{NER}$ reports.

Nonetheless, the overall performance of these DAPT models should also be assessed in light of the total dataset size and the distribution of entities in the task. Performing cross-validation is also desirable for more sound results. This study did not intend to maximize these models' performance; hence, an investigation into the best set of hyperparameters, as well as tagging notation, could greatly improve their performance.

5 CONCLUSION

We introduce YACSDB$_{NER}$, a publicly available dataset for Named Entity Recognition in the cybersecurity domain, structured around the STIX language. The dataset comprises 24,878 sentences from 422 cybersecurity reports in IOB format. Design choices for the dataset were based on a comprehensive review of state-of-the-art architectures for the Named Entity Recognition task for many different goals in cybersecurity. YACSDB$_{NER}$ is designed for the extraction of STIX Domain Objects task, enabling easy CTI sharing and supporting the construction of knowledge graphs.

Our dataset addresses key limitations observed in existing datasets by having up-to-date reports and a STIX common language. In addition, we describe a semi-automated annotation pipeline with Gemini that leverages a fine-tuned autoregressive LLM combined with a fine-tuned BERT model on an iterative labeling process. This process enabled the annotation of a cybersecurity corpus using a few annotators, which can reduce annotation costs and promote community-driven dataset enhancement. Lastly, YACSDB$_{NER}$ addresses the lack of benchmarks for NER models, which made models hard to compare, therefore supporting the development of new models as well as their evaluation.

The dataset is applied for fine-tuning seven BERT-based models. By doing this, it is possible to observe its effectiveness in the task it was built for: cybersecurity NER. Results support the use of YACSDB$_{NER}$ as a benchmark. A deeper analysis of DAPT BERT models in the cybersecurity domain is deemed necessary, as most of them had suboptimal outcomes. It was also noted that the SecureBERT domain adaptation strategy benefited the most from our dataset. YACSDB$_{NER}$ advances the landscape of NER datasets in the cybersecurity domain, enabling tasks such as the extraction of structured knowledge graphs from unstructured text.

5.1 FUTURE WORKS

Many opportunities unfold from the results. For future work, we propose to further extend the labeling process to the remainder of the reports and investigate the impact of the dataset size on the performance of the models. Furthermore, the dataset would greatly benefit from comparing it to the existing datasets and defining points of improvement in NER tasks.

With this foundational research, a diagnostic study will be undertaken to investigate the suboptimal performance observed in DAPT models, encompassing a comparative analysis with other Transformer-based architectures beyond BERT. Moreover, the development of strategies that leverage the available public datasets jointly will be explored.

BIBLIOGRAPHY

1 JOINT Publication 2-0 Joint Intelligence. *US Department o Defense*, 2007.

2 CHISMON, D.; RUKS, M. Threat intelligence: Collecting, analysing, evaluating. *MWR InfoSecurity Ltd*, v. 3, n. 2, p. 36–42, 2015.

3 CALTAGIRONE, S.; PENDERGAST, A.; BETZ, C. The diamond model of intrusion analysis. *Threat Connect*, v. 298, n. 0704, p. 1–61, 2013.

4 ALPAYDIN, E. *Introduction to Machine Learning, fourth edition*. [S.l.]: MIT Press, 2020. (Adaptive Computation and Machine Learning series). ISBN 9780262043793.

5 VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER Łukasz; POLOSUKHIN, I. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. v. 2017-December. ISSN 10495258.

6 ALAMMAR, J. *Inteligência*. 2020. Last access 2025-05-03. Disponível em: <https://www.gov.br/abin/pt-br/assuntos/inteligencia-e-contrainteligencia/inteligencia>.

7 DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. [S.l.: s.n.], 2019. p. 4171–4186.

8 JORDAN, B.; PIAZZA, R.; DARLEY, T. *STIX Version 2.1*. 2021. <https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html>. [Accessed 28-02-2025].

9 SAUERWEIN, C.; PFOHL, A. Towards automated classification of attackers' ttps by combining nlp with ml techniques. *arXiv preprint arXiv:2207.08478*, 2022.

10 Recorded Future. *The Intelligence Handbook: A Roadmap for Building an Intelligence-Led Security Program*. Fourth edition. CyberEdge Group: Annapolis, MD, USA, 2022. Last access 2025-05-09. Disponível em: <https://go.recordedfuture.com/the-intelligence-handbook-fourth-edition>.

11 ZHAO, J.; YAN, Q.; LI, J.; SHAO, M.; HE, Z.; LI, B. TIMiner: Automatically extracting and analyzing categorized cyber threat intelligence from social data. *Computers and Security*, v. 95, 2020. ISSN 01674048.

12 RANADE, P.; PIPLAI, A.; JOSHI, A.; FININ, T. Cybert: Contextualized embeddings for the cybersecurity domain. In: *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021*. [S.l.: s.n.], 2021.

13 RAHMAN, M. R.; MAHDAVI-HEZAVEH, R.; WILLIAMS, L. A literature review on mining cyberthreat intelligence from unstructured texts. In: *IEEE International Conference on Data Mining Workshops, ICDMW*. [S.l.: s.n.], 2020. v. 2020-November. ISSN 23759259.

14 BELTAGY, I.; LO, K.; COHAN, A. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

15 LEE, J.; YOON, W.; KIM, S.; KIM, D.; KIM, S.; SO, C. H.; KANG, J. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, Oxford University Press, v. 36, n. 4, p. 1234–1240, 2020.

16 PENG, B.; CHERSONI, E.; HSU, Y. Y.; HUANG, C. R. Is domain adaptation worth your investment? comparing bert and finbert on financial tasks. In: *Proceedings of the 3rd Workshop on Economics and Natural Language Processing, ECONLP 2021*. [S.l.: s.n.], 2021. Domain adaptation shows to be good on biomedical domain.

17 AGHAEI, E.; NIU, X.; SHADID, W.; AL-SHAER, E. Securebert: A domain-specific language model for cybersecurity. In: *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*. [S.l.: s.n.], 2023. v. 462 LNICST. ISSN 1867822X.

18 WANG, X.; HE, S.; XIONG, Z.; WEI, X.; JIANG, Z.; CHEN, S.; JIANG, J. Aptner: A specific dataset for ner missions in cyber threat intelligence field. In: *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2022*. [S.l.: s.n.], 2022.

19 BAYER, M.; KUEHN, P.; SHANEHSAZ, R.; REUTER, C. Cysecbert: A domain-adapted language model for the cybersecurity domain. *ACM Transactions on Privacy and Security*, v. 27, 2024. ISSN 24712574.

20 AMERI, K.; HEMPEL, M.; SHARIF, H.; LOPEZ, J.; PERUMALLA, K. Cybert: Cybersecurity claim classification by fine-tuning the bert language model. *Journal of Cybersecurity and Privacy*, v. 1, 2021. ISSN 2624800X.

21 MARCHIORI, F.; CONTI, M.; VERDE, N. V. Stixnet: A novel and modular solution for extracting all stix objects in cti reports. In: *ACM International Conference Proceeding Series*. [S.l.: s.n.], 2023.

22 SIRACUSANO, G.; SANVITO, D.; GONZALEZ, R.; SRINIVASAN, M.; KAMATCHI, S.; TAKAHASHI, W.; KAWAKITA, M.; KAKUMARU, T.; BIFULCO, R. Time for action: Automated analysis of cyber threat intelligence in the wild. *arXiv preprint arXiv:2307.10214*, 2023.

23 YI, F.; JIANG, B.; WANG, L.; WU, J. Cybersecurity named entity recognition using multi-modal ensemble learning. *IEEE Access*, v. 8, 2020. ISSN 21693536.

24 Agência Brasileira de Inteligência (ABIN). *Inteligência*. 2020. Last access 2025-05-03. Disponível em: <https://www.gov.br/abin/pt-br/assuntos/inteligencia-e-contrainteligencia/inteligencia>.

25 LOWENTHAL, M. M.; CLARK, R. M. *The five disciplines of intelligence collection*. [S.l.]: Sage, 2015.

26 BLOCK, L. The long history of osint. *Journal of Intelligence History*, Taylor & Francis, v. 23, n. 2, p. 95–109, 2024.

27 CROWTHER, G. A. The cyber domain. *The Cyber Defense Review*, Army Cyber Institute, v. 2, n. 3, p. 63–78, 2017. ISSN 24742120, 24742139. Disponível em: <http://www.jstor.org/stable/26267386>.

28 DIMITRIJEVIĆ, I. R.; MANDIĆ, G. J. Open sources of data and intelligence. *Bezbednost, Beograd*, v. 64, n. 2, p. 80–98, 2022.

29 PASTOR-GALINDO, J.; NESPOLI, P.; MÁRMOL, F. G.; PÉREZ, G. M. The not yet exploited goldmine of osint: Opportunities, open challenges and future trends. *IEEE access*, IEEE, v. 8, p. 10282–10304, 2020.

30 ALSMADI, I. Cyber intelligence analysis. In: *The NICE Cyber Security Framework: Cyber Security Intelligence and Analytics*. [S.l.]: Springer, 2023. p. 85–129.

31 SHIN, J.-H.; CHEON, S.-P.; EOM, J. The role and responsibility of cyber intelligence in cyber warfare. *Advanced Science and Technology Letters*, v. 51, p. 305–308, 2014.

32 FRIEDMAN, J.; BOUCHARD, M. Definitive guide to cyber threat intelligence. *Friedman, M. Bouchard.-2015*, 2015.

33 WANG, G.; LIU, P.; HUANG, J.; BIN, H.; WANG, X.; ZHU, H. Knowcti: Knowledge-based cyber threat intelligence entity and relation extraction. *Computers & Security*, Elsevier, v. 141, p. 103824, 2024.

34 ZHOU, Y.; TANG, Y.; YI, M.; XI, C.; LU, H. Cti view: Apt threat intelligence analysis system. *Security and Communication Networks*, Wiley Online Library, v. 2022, n. 1, p. 9875199, 2022.

35 HUTCHINS, E. M.; CLOPPERT, M. J.; AMIN, R. M. et al. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, v. 1, n. 1, p. 80, 2011.

36 STROM, B. E.; APPLEBAUM, A.; MILLER, D. P.; NICKELS, K. C.; PENNINGTON, A. G.; THOMAS, C. B. Mitre att&ck - design and philosophy. *Https://Www.Mitre.Org/Sites/Default/Files/Publications/Pr-18-0944-11-Mitre-Attack-Design-and-Philosophy.Pdf*, 2018.

37 KIRILLOV, I.; BECK, D.; CHASE, P.; MARTIN, R. Malware attribute enumeration and characterization. *The MITRE Corporation [online, accessed Apr. 8, 2019]*, 2011.

38 MITRE Corporation. *CAPEC - Common Attack Pattern Enumeration and Classification*. 2023. <https://capec.mitre.org/>. Accessed: 2025-04-09.

39 MITRE Corporation. *Cyber Observable eXpression — CybOX™: A Structured Language for Cyber Observables*. 2020. <https://measurablesecurity.mitre.org/docs/cybox-intro-handout.pdf>. Accessed: 2025-04-09.

40 KRAUSS, O.; PAPESH, K. Analysis of threat intelligence information exchange via the stix standard. In: IEEE. *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. [S.l.], 2022. p. 1–6.

41 WEAVER, W. Translation. In: *Proceedings of the conference on mechanical translation*. [S.l.: s.n.], 1952.

42 HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT press, v. 9, n. 8, p. 1735–1780, 1997.

43 CHURCH, K. W. Word2vec. *Natural Language Engineering*, Cambridge University Press, v. 23, n. 1, p. 155–162, 2017.

44 PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543.

45 JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; MIKOLOV, T. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

46 PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. Disponível em: <http://arxiv.org/abs/1802.05365>.

47 RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. et al. *Improving language understanding by generative pre-training.(2018)*. 2018.

48 TAYLOR, W. L. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, SAGE Publications Sage CA: Los Angeles, CA, v. 30, n. 4, p. 415–433, 1953.

49 DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. Disponível em: <https://arxiv.org/abs/1810.04805v1>.

50 LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

51 RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. et al. Language models are unsupervised multitask learners. *OpenAI blog*, v. 1, n. 8, p. 9, 2019.

52 SUN, C.; QIU, X.; XU, Y.; HUANG, X. How to fine-tune bert for text classification? In: SPRINGER. *China national conference on Chinese computational linguistics*. [S.l.], 2019. p. 194–206.

53 EPURE, E. V.; HENNEQUIN, R. Probing pre-trained auto-regressive language models for named entity typing and recognition. *arXiv preprint arXiv:2108.11857*, 2021.

54 CHOWDHERY, A.; NARANG, S.; DEVLIN, J.; BOSMA, M.; MISHRA, G.; ROBERTS, A.; BARHAM, P.; CHUNG, H. W.; SUTTON, C.; GEHRMANN, S. et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, v. 24, n. 240, p. 1–113, 2023.

55 AL, G. T. et. *Gemini: A Family of Highly Capable Multimodal Models*. 2024. Disponível em: <https://arxiv.org/abs/2312.11805>.

56 HENDRYCKS, D.; BURNS, C.; BASART, S.; ZOU, A.; MAZEIKA, M.; SONG, D.; STEINHARDT, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

57 AL, G. T. et. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. 2024. Disponível em: <https://arxiv.org/abs/2403.05530>.

58 WANG, L.; LI, R.; YAN, Y.; YAN, Y.; WANG, S.; WU, W.; XU, W. Instructionner: A multi-task instruction-based generative framework for few-shot ner. *arXiv preprint arXiv:2203.03903*, 2022.

59 HUSARI, G.; AL-SHAER, E.; AHMED, M.; CHU, B.; NIU, X. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In: *ACM International Conference Proceeding Series*. [S.l.: s.n.], 2017. Part F132521.

60 YOU, Y.; JIANG, J.; JIANG, Z.; YANG, P.; LIU, B.; FENG, H.; WANG, X.; LI, N. Tim: threat context-enhanced ttp intelligence mining on unstructured threat data. *Cybersecurity*, v. 5, 2022. ISSN 25233246.

61 ALVES, P.; FILHO, G. P. R.; GONÇALVES, V. P. Modelo de classificação de TTP baseado em transformadas BERT. *Proceedings of the Ibero American Conferences on Applied Computing 2022 and WWW/Internet 2022*, 2022. Disponível em: <https://www.iadisportal.org/digital-library/modelo-de-classifica%C3%A7%C3%A3o-de-ttp-baseado-em-transformadas-bert>.

62 TIKHOMIROV, M.; LOUKACHEVITCH, N.; SIROTINA, A.; DOBROV, B. Using bert and augmentation in named entity recognition for cybersecurity domain. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2020. v. 12089 LNCS. ISSN 16113349.

63 LIM, S. K.; MUIS, A. O.; LU, W.; ONG, C. H. Malwaretextdb: A database for annotated malware articles. In: *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. [S.l.: s.n.], 2017. v. 1.

64 LEGOY, V.; CASELLI, M.; SEIFERT, C.; PETER, A. *Automated Retrieval of ATT&CK Tactics and Techniques for Cyber Threat Reports*. 2020.

65 DING, N.; XU, G.; CHEN, Y.; WANG, X.; HAN, X.; XIE, P.; ZHENG, H. T.; LIU, Z. Few-nerd: A few-shot named entity recognition dataset. In: *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. [S.l.: s.n.], 2021.

66 DASGUPTA, S.; PIPLAI, A.; KOTAL, A.; JOSHI, A. A comparative study of deep learning based named entity recognition algorithms for cybersecurity. In: IEEE. *2020 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2020. p. 2596–2604.

67 WANG, X.; LIU, X.; AO, S.; LI, N.; JIANG, Z.; XU, Z.; XIONG, Z.; XIONG, M.; ZHANG, X. Dnrti: A large-scale dataset for named entity recognition in threat intelligence. In: IEEE. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. [S.l.], 2020. p. 1842–1848.

68 EVANGELATOS, P.; ILIOU, C.; MAVROPOULOS, T.; APOSTOLOU, K.; TSIKRIKA, T.; VROCHIDIS, S.; KOMPATSIARIS, I. Named entity recognition in cyber threat intelligence using transformer-based models. In: *Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience, CSR 2021*. [S.l.: s.n.], 2021.

69 HANKS, C.; MAIDEN, M.; RANADE, P.; FININ, T.; JOSHI, A. et al. Recognizing and extracting cybersecurity entities from text. In: *Workshop on Machine Learning for Cybersecurity, International Conference on Machine Learning*. [S.l.: s.n.], 2022.

70 DEKA, P.; RAJAPAKSHA, S.; RANI, R.; ALMUTAIRI, A.; KARAFILI, E. Attacker: towards enhancing cyber-attack attribution with a named entity recognition dataset. In: SPRINGER. *International Conference on Web Information Systems Engineering*. [S.l.], 2024. p. 255–270.

71 KONKOL, M.; KONOPÍK, M. Segment representations in named entity recognition. In: SPRINGER. *International conference on text, speech, and dialogue*. [S.l.], 2015. p. 61–70.

72 BAYER, M.; KAUFHOLD, M. A.; REUTER, C. A survey on data augmentation for text classification. *ACM Computing Surveys*, v. 55, 2022. ISSN 15577341.

73 HAN, Z.; GAO, C.; LIU, J.; ZHANG, J.; ZHANG, S. Q. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

74 HU, E. J.; SHEN, Y.; WALLIS, P.; ALLEN-ZHU, Z.; LI, Y.; WANG, S.; WANG, L.; CHEN, W. et al. Lora: Low-rank adaptation of large language models. *ICLR*, v. 1, n. 2, p. 3, 2022.

75 CASCANTE-BONILLA, P.; TAN, F.; QI, Y.; ORDONEZ, V. Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning. In: *Proceedings of the AAAI conference on artificial intelligence*. [S.l.: s.n.], 2021. v. 35, n. 8, p. 6912–6920.

76 GHOSH, K.; BELLINGER, C.; CORIZZO, R.; BRANCO, P.; KRAWCZYK, B.; JAPKOWICZ, N. The class imbalance problem in deep learning. *Machine Learning*, Springer, v. 113, n. 7, p. 4845–4901, 2024.

77 ZANELLA, L.; TOUSSAINT, Y. Adding linguistic information to transformer models improves biomedical event detection? In: IEEE. *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. [S.l.], 2023. p. 1211–1216.

APPENDIX

I. STIX BUNDLE EXAMPLE

Listing I.1: STIX bundle from Figure 2.4

```
{
  "type": "bundle",
  "id": "bundle--49728cb0-964f-4300-b6ff-4d5fd5cf147c",
  "spec_version": "2.1",
  "objects": [
    {
      "type": "marking-definition",
      "id": "marking-definition--e7e6db3f-3a54-4982-a723-09cb78d8bc91",
      "created": "2025-05-25T00:00:00Z",
      "definition_type": "statement",
      "definition": {
        "statement": "STIX Bundle Example. "
      }
    },
    {
      "type": "malware",
      "spec_version": "2.1",
      "id": "malware--ce59d4e9-96a2-4d98-92db-132b82e6c671",
      "created": "2025-05-25T00:00:00Z",
      "modified": "2025-05-25T00:00:00Z",
      "name": "Guildma payload",
      "malware_types": [
        "remote-access-trojan"
      ],
      "is_family": false,
      "sample_refs": [
        "file--fbc452e0-1469-41ac-bbcb-00f9f04f37a7"
      ]
    },
    {
      "type": "malware-analysis",
      "spec_version": "2.1",
      "id": "malware-analysis--8749378b-9e64-4a63-a291-b059d7294e88",
      "created": "2025-05-25T00:00:00Z",
      "modified": "2025-05-25T00:00:00Z",
      "product": "MadeUpAnalyzer",
      "analysis_engine_version": "6.4.2",
      "analysis_definition_version": "20250525-001",
      "analysis_started": "2025-05-25T00:00:00Z",
      "analysis_ended": "2025-05-25T00:00:00Z",
      "result": "malicious"
    },
    {
      "type": "relationship",
```

```
      "spec_version": "2.1",
      "id": "relationship--b9b4907f-fd6e-43c7-8a3a-397fbce7ad0e",
      "created": "2025-05-25T00:00:00Z",
      "modified": "2025-05-25T00:00:00Z",
      "relationship_type": "analysis-of",
      "source_ref": "malware-analysis--8749378b-9e64-4a63-a291-b059d7294e88",
      "target_ref": "malware--ce59d4e9-96a2-4d98-92db-132b82e6c671"
    },
    {
      "type": "file",
      "spec_version": "2.1",
      "id": "file--fbc452e0-1469-41ac-bbcb-00f9f04f37a7",
      "hashes": {
        "MD5": "a92e5b2bae0b4b3a3d81c85610b95cd4",
        "SHA-1": "5374e08903744ceeaedd8f5e1bfc06b2c4688e76"
      },
      "size": 77312,
      "name": "guildma.exe",
      "parent_directory_ref": "directory--38545efb-e159-4e44-943b-06a71a2f7b93"
    },
    {
      "type": "directory",
      "spec_version": "2.1",
      "id": "directory--38545efb-e159-4e44-943b-06a71a2f7b93",
      "path": "C:\\Program Files\\Diebold\\Warsaw\\"
    }
  ]
}
```