SoK: Automated TTP Extraction from CTI Reports – Are We There Yet?

Marvin Büchel[†,1], Tommaso Paladini[†,2,6], Stefano Longari[2], Michele Carminati[2], Stefano Zanero[2], Hodaya Binyamini[4], Gal Engelberg[4], Dan Klein[4], Giancarlo Guizzardi[3], Marco Caselli[5], Andrea Continella[3], Maarten van Steen[3], Andreas Peter[1], and Thijs van Ede[3]

[1]Carl von Ossietzky Universität Oldenburg, [2]Politecnico di Milano, [3]University of Twente, [4]Accenture Labs, [5]Siemens AG, [6]NEC Laboratories Europe

Abstract

Cyber Threat Intelligence (CTI) plays a critical role in sharing knowledge about new and evolving threats. With the increased prevalence and sophistication of threat actors, intelligence has expanded from simple indicators of compromise to extensive CTI reports describing high-level attack steps known as Tactics, Techniques and Procedures (TTPs). Such TTPs, often classified into the ontology of the ATT&CK framework, make CTI significantly more valuable, but also harder to interpret and automatically process. Natural Language Processing (NLP) makes it possible to automate large parts of the knowledge extraction from CTI reports; over 40 papers discuss approaches, ranging from named entity recognition over embedder models to generative large language models. Unfortunately, existing solutions are largely incomparable as they consider decisively different and constrained settings, rely on custom TTP ontologies, and use a multitude of custom, inaccessible CTI datasets. We take stock, systematize the knowledge in the field, and empirically evaluate existing approaches in a unified setting for fair comparisons. We gain several fundamental insights, including (1) the finding of a kind of performance limit that existing approaches seemingly cannot overcome as of yet, (2) that traditional NLP approaches (possibly counterintuitively) outperform modern embedder-based and generative approaches in realistic settings, and (3) that further research on understanding inherent ambiguities in TTP ontologies and on the creation of qualitative datasets is key to take a leap in the field.

1 Introduction

In today's rapidly evolving cyber landscape, the sophistication and frequency of cyber threats are increasingly growing, challenging the ability of cybersecurity teams to anticipate, identify, and respond to new threats. Cyber Threat Intelligence (CTI) is the collection, analysis, and use of information about current and potential threats in the cyber landscape to help organizations detect, prevent, and respond to attacks. A central component of CTI is the understanding of Tactics, Techniques, and Procedures (TTPs), which describe the high-level goals (*tactics*) of adversaries, the way they attempt to achieve them (*techniques*), and the specific methods or processes they employ to carry out their objectives (*procedures*). Unfortunately, information on TTPs is often only given in unstructured form as part of human-written textual *CTI reports*, which must be manually analyzed and extracted. As the volume and complexity of CTI reports increase, management, structuring, and timely extraction of valuable knowledge from these reports has become a critical task for preventive and reactive measures in threat scenarios [92]. The MITRE ATT&CK framework [89] has emerged as the de facto standard for structuring TTPs from CTI reports, offering a comprehensive knowledge base and *ontology* of adversarial tactics and techniques from real-world observations that are widely recognized by the cybersecurity community. Such a knowledge base provides security professionals with a common language for understanding, categorizing, and mitigating cyber threats. By mapping CTI reports to the TTPs in this framework, organizations can effectively use this information to better anticipate, detect, and respond to threats [4]. Manually mapping CTI reports to TTPs, however, is an error-prone task that requires security expertise and human resources, which many organizations cannot allocate. Hence, a lot of research deals with the automation of this knowledge extraction process. Existing solutions are based on Natural Language Processing (NLP) techniques, ranging from Named Entity Recognition (NER) [28, 38, 58, 59, 85] to data-driven classification [37, 68, 72, 77, 79] and generative Large Language Models (LLMs) [14, 15, 24, 26, 88], all of which, when adapted to the CTI domain, are shown to be promising in automatically extracting TTPs from CTI reports.

As the field advances, however, the diverse landscape of NLP techniques for TTP extraction has become increasingly complex. Researchers have proposed numerous approaches, each with distinct benefits and limitations. Existing solutions turn out to be largely incomparable as they are often tai-

[†]Authors contributed equally to this work.

lored to specific constrained settings, relying on custom TTP ontologies and (confidential) CTI datasets, sometimes even using different evaluation metrics (see Section 2 for a detailed analysis). This diversity and incomparability hinder a comprehensive understanding of the current state of the field.

To address this problem, we contribute with:

- a **Systematization** of all existing NLP-based solutions for knowledge extraction from CTI reports (i.e., the state-of-the-art in automatic TTP extraction) based on a detailed literature analysis; we classify the existing works along different dimensions, identify their benefits and limitations, and provide guidelines and future directions,
- a **Technical Implementation** of the different NLP approaches for in-depth comparison and for fostering of future research on security-specific NLP challenges, and
- an **Empirical Evaluation** of the existing NLP approaches, including emerging ones such as (generative) LLMs, in a unified setting enabling fair comparisons.

Key insights of our research include

1. even when considering only the top-50 most common TTPs, the best-performing solutions reach a precision of around 80% for a recall of around 66%; for less common TTPs, performances quickly drop (esp. for precision, going way below 40% in realistic settings),
2. possibly counterintuitively, in a scenario that replicates real-world settings, traditional NLP approaches still notably outperform modern embedder-based and generative NLP approaches, and
3. while the literature focuses on developing evermore novel NLP-based technologies to improve performances, our results strongly suggest that further research efforts on understanding inherent ambiguities in used ontologies as well as on the creation of large, qualitative datasets are urgently needed to take a leap in the field.

2 Systematization

We systematize existing work into three overarching classes of NLP approaches used for TTP extraction based on their NLP objective. First, we discuss **Named Entity Recognition** (**NER**), which aims to identify explicitly mentioned entities in the text. Second, we look at **Classification** approaches that can deal with both explicit and implicit references but require more (often labeled) data for training, which is scarce in the CTI domain. Finally, we look at **Generative LLMs** approaches that are adapted to the domain of TTP extraction.

2.1 Literature Collection

To collect literature, we searched the most common search platforms for security-related works DBLP[1] and Google Scholar[2] using the query *("[cyber] threat intelligence" OR "CTI" OR "TTP" OR "att&ck" OR "threat report" OR ("attack" AND ("pattern" OR "technique" OR "behavior" OR "graph")) AND "extraction" OR "mining" OR "classification")*. Here, we focused on the inclusion of threat intelligence, as well as attack techniques and the extraction thereof. To limit the search to relevant papers, we only searched for works published after 2015, the release of the MITRE's ATT&CK framework, and limited the search to 100 entries for each keyword, resulting in a total of 1223 papers. Thereafter, we performed a manual evaluation of the titles and abstracts to exclude papers outside of our field of study, resulting in 440 relevant works. Next, we excluded papers that (1) do not propose an approach for extracting TTPs, and (2) do not process CTI reports as the input of the TTP processing pipeline. To expand the final search, we performed a citation-based search [46], which led to finding a total of 38 relevant papers and 2 blog articles [82, 99]. While we acknowledge that this literature collection may not be complete, the resulting papers cover a wide range of techniques used in the field that are representative of the state-of-the-art in TTP extraction.

Related Works. CTI literature discusses approaches for different purposes than TTP extraction. NER approaches are often used to extract CTI keywords from online discussions [23, 73], or to construct cybersecurity knowledge graphs [74]. Classification approaches are also developed to classify Indicators of Compromise (IoCs) [67, 109], perform topic modeling [21], identify threat actors [70], or collect CTI-relevant documents from unstructured collections [43]. The more recent Generative approaches are proposed for the development of CTI "AI-assistants" [40] or for the identification of relevant CTI [87]. We exclude these from our analysis, as we limit ourselves purely to TTP extraction.

2.2 Named Entity Recognition (NER)

NER approaches use a combination of rule-based and ML-based techniques to process the semantical meaning of a text and thereby identify explicitly mentioned entities, in our case TTPs. The advantage is that NER clearly indicates where a specific TTP is mentioned in a CTI report and that it allows a high level of manual control over which TTPs are detected as rules can easily be adapted. The disadvantage is that these methods cannot detect implicit references to TTPs and are often less flexible when processing text that is phrased differently from what is expected.

[1] https://dblp.org
[2] https://scholar.google.com/

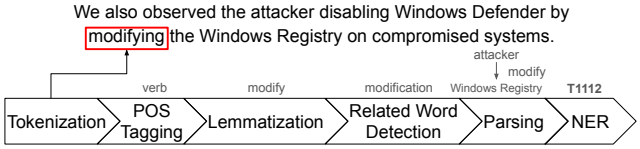

Figure 1: Generic Named Entity Recognition (NER) pipeline.

2.2.1 Methods

Figure 1 shows the different steps that are taken by various NER approaches to TTP extraction from CTI reports. We note that different NER approaches do not necessarily employ all steps and often customize the rules and implementations used by each step. The first step of all NER approaches is to tokenize the text to distinguish between punctuation, individual words[3] (referred to as tokens), and to separate sentences, which allows subsequent processing of the text [95]. Next, tokens are tagged with their part-of-speech (POS), indicating whether words are nouns, verbs, adjectives, etc. [4] [64]. These POS tags allow NER approaches to differentiate between words with different meanings, e.g., in the example of Figure 1 "attacks" refers to the plural of the noun "attack" rather than the action (i.e., verb) "to attack". POS tags also improve lemmatization approaches, which aim to reduce different variations of the same word to their base form that can be used to recognize named entities later on [8, 76]. E.g., in the given example "executing" should be detected when searching for "execute" as they are different variations of the same verb. However, lemmatization is limited to tokens with the same POS tag and base form; it cannot find words with similar meaning (e.g., synonyms). Therefore, various approaches include related word detection [25] that allows NER to find e.g., "execute" (verb) when searching for "execution" (noun). Next, some approaches leverage parsing, which identifies the sentence subject, objects, and other relations within the text [19, 48, 71]. Parsing is required when searching for named entities consisting of multiple tokens or even entire sentences, e.g., matching "escalation of privileges" when searching for "privilege escalation". Moreover, parsing can be used to identify relations between extracted TTPs, but this is considered outside of the scope of this work. Finally, NER leverages these preprocessing steps to identify named entities using either rules or ML for classifying individual tokens into named entities [54].

State-of-the-art. Using these techniques, we classify the state-of-the-art in NER-based TTP extraction in Table 1. Here, we make a distinction between semantic approaches that use rules for extracting TTPs and hybrid approaches that locate named entities in the text (i.e., perform NER), but do so by machine-learning classification on individual tokens in the text. We note that we still categorize these techniques as NER because they solely focus on explicitly mentioned entities and do not find implicit references. From this table, we find that approaches leverage different subsets of discussed techniques making it difficult to assess the influence of individual steps. Moreover, all approaches are tested on a different dataset and are labeled with a different ontology making comparison of the approaches difficult. In Section 4, we will investigate the individual NER methods and evaluate to which extent they contribute to TTP extraction.

Table 1: Components implemented by related work.

Type	Approach	Tok.	POS	Lem.	Rel.	Par.	NER	Data	Ontology
Semantic	AttacKG [58]	❍	❍	❍	×	❍	●	Custom	Custom
	Extractor [85]	●	●	●	❍	●	●	[20, 66]	Custom
	TTPDrill [38]	●	●	×	●	❍	●	Custom	Custom
Hybrid	ActionMiner [39]	❍	❍	×	×	❍	●	Custom	Custom
	CASIE [86]	❍	❍	❍	×	●	●	Custom	Custom
	CyberEntRel [3]	❍	×	×	❍	×	●	Custom	Custom
	EX-Action [103]	●	❍	×	×	❍	●	Custom	ATT&CK
	Ghazi et al. [32]	❍	❍	×	×	×	❍	Custom	Custom
	ThreatKG [27]	●	×	×	×	●	●	Custom	Custom
	TIMiner [106]	❍	×	❍	●	❍	●	Custom	Custom

× action is not present or not mentioned.
❍ action is present, but not domain-specific.
● action is present and domain-specific.

Tok. = Tokenization
POS = Part-of-Speech
Lem. = Lemmatization
Rel. = Related word detection
Par. = Parsing
NER = Named Entity Recognition

2.3 Classification

As the name suggests, classification approaches process CTI reports and classify them into a set of predetermined TTPs. Reports can be classified in their entirety, per sentence, or even per token (see Section 3). Initial works split text into tokens similar to NER, converted them to an unordered set (known as a Bag-of-Words), and extracted features such as the Term Frequency (TF), specifying the occurrences of each term (i.e., token), or Term Frequency-Inverse Document Frequency (TF-IDF) [42], which discounts generic tokens that relate to many classes and promotes specific tokens related to a specific class. Subsequently, these works feed the extracted features into a machine learning model (e.g., SVM) to perform classification [7, 52, 61]. Later works transform tokens, sentences or full pieces of text into a vector representation called embeddings [30, 31, 52]. Word2Vec [65] learned token representations by analyzing surrounding tokens in unlabeled pieces of text, whereas other works such as BERT [22] improved upon these embeddings by leveraging attention networks to find relations between tokens. The idea is that these embeddings encapsulate the meaning of text by not analysing the individual token, but the entire context. Subsequently, these embeddings can be compared directly with embeddings of target classes in an unlabeled manner or be fed to a classifier if the text is labeled. Moreover, the embeddings can even be fine-tuned for classification when neural network layers are used for classification.

[3] Sometimes tokenization uses multiple tokens to split up words outside its predefined vocabulary (e.g., "executing" becomes "execute" + "-ing").

[4] See https://universaldependencies.org/u/pos/ for a full list of POS tags.

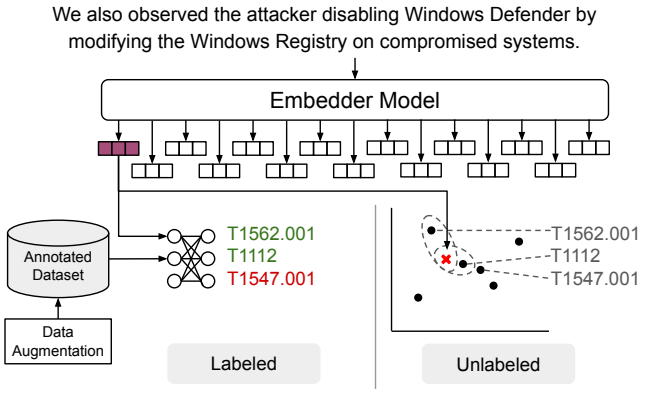

Figure 2: Generic classification model pipeline using either labeled data to train a classification network on top of embeddings, or directly compares the embeddings with embeddings of target classes in an unlabeled approach.

2.3.1 Methods

In this section, we focus on the more recent embedder-based classification as the majority of state-of-the-art research leverages this approach. In addition, Orbinato et al. [72] perform a wide analysis on this task by comparing classifiers relying on traditional NLP techniques [7, 30, 52] and transformers encoders, observing that the latter achieve the most competitive results. Thus, Figure 2 shows the pipeline for classifying TTPs from CTI documents using embeddings which are obtained from a (pre-trained) encoder. These embeddings are either used to train a classifier that requires labeled data (left), or can be used in a Semantic Text Similarity (STS) approach that compares text to descriptions of the target class, e.g., descriptions of ATT&CK techniques (right).

Embedder model. The model used to generate the text embeddings plays a large role in classification performance as embeddings that do not capture the semantic meaning of a text will be useless for predicting relevant TTPs. Subsequently, other models such as RoBERTa [62] have suggested improvement in the training process and use larger datasets to create better generic embeddings. However, such models have not been trained to handle the terminology of domain-specific texts such as CTI reports [75]. Therefore, models such as SecBERT [60], and CTI-BERT [75] have been pre-trained from scratch on cybersecurity texts. Moreover, with continual pre-training [44], already pre-trained models can be adapted for a domain-specific task, resulting in works such as SecureBERT [2], CyBERT [78], DarkBERT [41], and CySecBERT [10]. Finally, models can be also fine-tuned for specific tasks by training them for a specific purpose, resulting in works such as Sentence-BERT (SBERT) [81] for semantic similarity computations, and consequent adaptations for cybersecurity such as ATTACK-BERT [1], and SentSecBERT [49]. For the sake of readability, we refer to

Table 2: Classification approaches for TTP extraction.

	Paper	Model	Augment. Art.	Augment. OOD	Mult.	Ont.	CTRs	Data	Gran.
Labeled	rcATT [52]	Word2Vec [65]	×	×	●	ATT.	●	Custom	Doc.
	Ayoade et al. [7]	None	×	×	●	ATT.	●	Custom	Doc.
	SeqMask [30], Ge et al. [31]	FastText [12]	×	×	●	ATT.	●	Custom	Doc.
	TRAM [82, 99]	SciBERT [11]	×	×	●	ATT.	●	[82]	Sent.
	Liu et al. [61]	Custom	×	×	●	ATT.	●	Custom	Doc.
	TTPHunter [79]	BERT-based [22]	×	×	×	ATT.	●	Custom	Sent.
	TIM [101]	SBERT [81]	×	×	×	ATT.	×	Custom	Sent.
	Tang et al. [91]	BERT [22]	×	×	×	ATT.	●	Custom	Sent.
	Alves et al. [6]	BERT-based [22, 84]	×	×	×	ATT.	×	Custom	Sent.
	Yan et al. [98]	BERT [22]	×	×	●	ATT.	●	Custom	Sent.
	Kim et al. [45]	None	●	×	●	ATT.	×	[82], Cust.	Doc.
	TTPXHunter [80]	SecureBERT [2]	●	×	×	ATT.	●	Custom	Sent.
	You et al. [100]	CTI-BERT [75]	×	●	●	ATT.	×	[82], Cust.	Sent.
	ALERT [77]	SciBERT [11]	×	×	×	ATT.	×	Custom	Sent.
	Li et al. [56]	BERT-based [22, 84]	×	×	●	ATT.	×	[82], Cust.	Sent.
	CTI-to-MITRE [72]	Multiple [60, 65]	×	×	●	ATT.	●	[82], Cust.	Both
	HMCAT [35]	SecureBERT [2]	●	×	×	ATT.	×	Custom	Both
	MITREtrieval [37]	RoBERTa [62]	×	×	●	ATT.	●	Custom	Sent.
	Fayyazi et al. [24]	BERT-based [2, 62]	×	×	●	ATT.	×	Custom	Sent.
Unlabeled	LADDER [5]	SBERT(MPNet) [81]	–	–	●	ATT.	×	Custom	Sent.
	Kumarasinghe et al. [49]	SentSecBERT [49]	–	–	●	ATT.	×	[82], Cust.	Sent.
	Abdeen et al. [1]	ATTACK-BERT [1]	–	–	●	ATT.	×	Custom	Sent.

Legend: (●) Element is present. (×) Element is not present or unclear from text. (–) Element does not apply. (Augment.) Data Augmentation. (Mult.) Multi-label. (Ont.) Ontology. (CTRs) TTP extraction from real-world CTI reports. (Gran.) Granularity. (Doc.) Document-level. (Sent.) Sentence-level.

all of the aforementioned pre-trained models as "CTI-specific models". In general, such CTI-specific models have been often shown to outperform other general language models on cybersecurity-related tasks [10, 73, 75], among which the extraction of TTPs from CTI reports [80].

Labeled approaches. Text classification models combine the embedding model with a feed forward network having a number of neurons equal to the number of classes. This consists in solving a multi-label text classification problem, where the input text is a single sentence. Even though BERT and its CTI-specific derivates are pre-trained, the chosen model should be nonetheless *fine-tuned* on the downstream NLP task with a labeled dataset. Fine-tuning requires finding optimal hyperparameters (i.e., non-trainable parameters), and several works have already studied optimal hyperparameter spaces, and the effects of different choices [6, 90]. This approach is largely employed in literature [6, 24, 37, 56, 61, 72, 77, 79, 80, 82, 98, 100]. In TTP extraction, the desired classifier should be able to provide 637 independent outputs – one for each technique and sub-technique – as this task is a multilabel classification problem, i.e., each sentence can be associated with multiple attack patterns. However, to the best of our knowledge, there is no public dataset that contains annotations for all the existing ATT&CK techniques and sub-techniques, posing a limitation to the capabilities of such an approach.

Data augmentation. As supervised machine learning models, labeled approaches suffer performance-related issues related to insufficient labeled data and class imbalance. In this context, datasets are scarcely available, and the distribution of TTPs in CTI documents is heavily skewed toward a small subset of techniques [100]. Training data can be augmented

with artificial samples to improve class representation and reduce overfitting, though it may introduce noise. Natural language data augmentation strategies typically consists in replicating sentences with text variations (e.g., synonyms, rephrasing) that preserve the meaning of the original sentence [96]. Some research works have adopted augmentation strategies that consist in generating synthetic samples, such as the Easy Data Augmentation (EDA) algorithm [45, 96], custom synonym substitution approaches based on Masked Word Prediction [55, 80], or text rephrasing with generative LLMs [35]. You et al. [100], instead, propose a method to augment training data with Out-of-Distribution (OOD) samples taken from the ATT&CK procedure descriptions and show that such auxiliary data is effective even if text styles differ.

Unlabeled approaches. Another approach solves TTP extraction as an information retrieval problem, such as Semantic Textual Similarity (STS), an NLP task that consists in measuring semantic equivalence between pairs of sentences [9]. In short, with a SBERT-based model [81], the similarity between input sentences embeddings and those of the ATT&CK TTP's descriptions can be calculated with cosine distance. Then, the sentences whose similarity exceeds a threshold can be assigned to the corresponding class. This threshold can be either treated as a simple hyperparameter and selected by minimizing the classification error on a validation set [5], or learned with a shallow logistic regression model [1]. Kumarasinghe et al. [49], instead, use this approach by chaining multiple models of increasing complexity for retrieving similar sentences to build a sentence annotation pipeline. Compared to labeled approaches, this approach has the advantage of not requiring a manually annotated dataset to fine-tune the model, but only to select the optimal threshold. Although this approach could, in principle, detect all possible TTP classes, Alam et al. [5] and Abdeen et al. [1] restrict their classifiers to a smaller subset, covering only 10.36% to 6.43% of the Enterprise matrix.

State-of-the-art. Table 2 presents related works following the text classification approach for TTP extraction. Most studies adopt the ATT&CK framework as the reference ontology and rely on labeled data despite the scarcity of labeled datasets, often creating custom datasets. The majority use sentence-level analysis, where the classifier input is a single sentence from a CTR. Some studies incorrectly treat the problem as a single-label multi-class classification task, limiting models to identifying only one TTP per sentence. However, prior research [5, 38] observed that individual sentences can describe multiple techniques. Only few works have focused on dataset imbalance issues and explored various data augmentation strategies to increase the performances of their text classifiers. Most works have opted for the inclusion of artificial data inside training data, while only one work shows the inclusion of OOD data in the training set. Note that data augmentation only applies to labeled approaches. BERT-based models, and, in particular, CTI-specific models are mostly chosen for building such classifiers, motivated by superior performances on the respective test datasets. In conclusion, we observe that comparing the results and methods from state-of-the-art research is challenging. This is mainly because most studies use custom datasets that are often publicly released. Additionally, text classifiers are usually tested on custom datasets containing only individual sentences, which does not fully reflect their ability to extract information from real-world reports.

2.4 Generation

Generative approaches leverage LLMs to produce new text, rather than classifying text into predetermined classes. The advantage of these models is that they often do not need to be fine-tuned on large datasets of labeled data, but need few to no examples [13, 47] or can be adapted using techniques discussed in Section 2.4.1. However, due to the novelty of these approaches there exist only a limited number works that use this approach (see Section 2.4.1).

2.4.1 Methods

Figure 3 gives a brief overview of the techniques in a generative LLM pipeline. While generative LLMs only require a prompt to produce results, additional methods have proven useful to increase performance and reduce hallucinations [13, 83]. We have categorised existing work according to the methods they use (Table 3) and find that often, only a subset of methods is used.

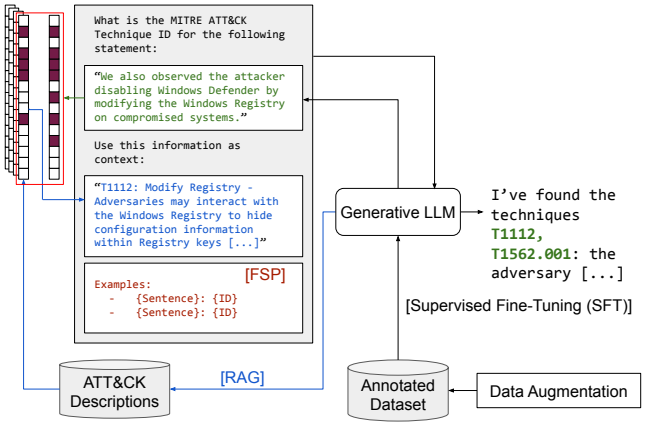

Figure 3: Generative Large Language Model (gLLM) pipeline.

Prompt Engineering. Prompt engineering is the process of optimizing input prompts for LLMs to improve task performance without modifying the model or its parameters. As a general rule, instructions should be clearly expressed, complex tasks should be broken down into smaller tasks, and the model should be helped to draw logical conclusions step by step with a chain-of-thought prompt [5]. Overall, there are

[5] https://platform.openai.com/docs/guides/prompt-engineering

Table 3: Generative approaches for TTP extraction.

Title	LLM Model	Ontology	SFT	FSP	RAG	CTRs	Doc.	Data
aCTIon [88]	GPT-3.5$_{\text{Turbo}}$	ATT&CK	×	×	×	●	×	Cust.
CTINexus [15]	GPT-4	MALOnt	×	●	●	●	●	Cust.
AECR [14]	ChatGLM3$_{6B}$	ATT&CK	●	×	×	●	×	Cust.
Fayyazi et al. [24]	GPT-3.5	ATT&CK	×	×	●	×	×	Cust.
Fengrui et al. [102]	Llama2	ATT&CK	●	×	×	×	×	Cust.
IntelEX [97]	GPT-4o$_{\text{mini}}$	ATT&CK	×	×	●	●	●	Cust.
Kumarasinghe et al. [49]	Multiple	ATT&CK	×	●	×	×	×	Cust.
AttacKG+ [105]	GLM-4	ATT&CK	×	×	×	●	×	Cust.
Fieblinger et al. [26]	Multiple	Custom	●	●	×	●	×	Cust.

Legend: (●) = Method is present, (×) = Method is not present, SFT = Supervised Fine-Tuning, FSP = Few-Shot Prompting, RAG = Retrieval-Augmented Generation, CTRs = TTP extraction from real-world CTI reports, Doc. = One-Shot inference of whole CTI report (in all stages). Data = dataset used for evaluation.

many different methodologies on how to customize prompts to either achieve a desired result or improve the performance of a specific task, which are explored in surveys [83].

Supervised Fine-Tuning. Supervised Fine-Tuning (SFT) of LLMs involves tailoring pre-trained models to specific tasks or domains using labeled datasets, refining their general-purpose knowledge to achieve specialized performance. By prioritizing domain-specific tasks, even small SFT models can beat those that are more than 10x larger [14]. SFT requires a sufficient amount of high-quality data and also sufficient computing resources for training. In addition, the training data must be transformed into natural language for training (e.g. question-answer pairs), which can also have an impact on the quality of the training.

In-Context Learning. In-Context learning is a way to provide the model with new knowledge at short-term after the pre-training, without having to further train the weights of the models using a large amount of data. In addition to the instruction, further context is added to the prompt, which can help to solve the problem or even contains the solution itself.

Few-Shot Prompting. Few-shot prompting is a method that provides examples of the task with suitable answers. The advantage is that the LLM not only gains a clearer understanding of what exactly needs to be searched for but also aligns its behavior, ensuring the output format and style match the desired specifications. Few-shot prompting can improve performance without much effort, as Brown et al. [13] already discovered in 2020. However, few-shot prompting can also lead to accuracy instability in the results, as Zhao et al. [107] demonstrate, the prompt format, the examples and even the order of the examples influence the effectiveness of few-shot prompting. This few-shot example bias was also observed when extracting CTI entities from Cheng et al. [15].

Retrieval-Augmented Generation. LLM have strong abilities to recall knowledge from the training process, but encounter challenges like being imprecise, having hallucination or outputting outdated knowledge in a difficult to understand black box manner. To prevent this, Retrieval-Augmented Generation (RAG) is an approach that helps generative models by retrieving relevant external data during inference, using semantic similarity to identify the most relevant information, without requiring new training. The retrieved knowledge is then used to enrich the prompt, with typically the top-k most relevant elements being added to provide the LLM with contextually accurate and up-to-date information for generating responses. As RAG has proven to be successful, it is developing very quickly and new methods are constantly being developed that are summarized in surveys [29].

State-of-the-art. Table 3 shows all the related-work that does TTP extraction with a generative LLM. It is evident that the existing works have employed a variety of methods, with no clear overarching trend emerging, except that each utilizes its own custom dataset for evaluation. It is noticeable that if supervised fine-tuning is in use [14, 102], then all other methods aimed at In-Context Learning (like FSP, RAG) are not used. With aCTIon [88], a generative LLM is used to summarize the content due to the small context size, so that another unmodified LLM can then predict all ATT&CK techniques at document level with a static prompt. CTINexus [15] use a three-level LLM Agent System to extract relationships between entities in addition to TTP extraction with dynamic RAG-like FSP. AECR [14] have fine-tuned a small 6 billion model by SFT to outperform much larger general models like GPT4 from OpenAI and to reduce hallucination by adding a linear head layer for classification. Advanced TTP Analysis [24] compared BERT-like models to generative LLMs with RAG, finding that BERT-like models had higher precision, while generative LLMs had higher recall, but were negatively impacted by incorrect RAG entries. In Few-Shot Learning of TTPs [102] the authors created augmented data with GPT4, which they used to train a LLama2-7B base model with SFT to successfully increase performance. IntelEX [97] combines RAG with a multi-agent system to increase precision. Kumarasinghe et al. [49] found that their multi-level classifier outperformed GPT-4, in terms of recall@3 metric with 0.81 against 0.44. AttacKG+ [105] uses a four-stage generative LLM pipeline to extract TTPs and entity relationships to build a knowledge graph. Fieblinger et al. [26] found that prompt engineering significantly impacts FSP performance, but its effectiveness varies greatly across different models. Meanwhile, SFT underperforms due to limited data availability.

2.5 Literature Takeaways

Most previous research has relied on labeled text classification approaches trained on annotated datasets, despite their limited availability, with some works focusing on methods to improve their performance (e.g., pre-training on CTI corpora, data augmentation). Initial approaches focused on the challenges related to the adaption of traditional NLP techniques, but rarely included methods specific to cybersecurity language. More recent approaches have explored novel generative models, due to their proved superior language understanding capabilities,

adopting and justifying different methods with no obvious common pattern. Overall, results are rarely demonstrated on public datasets, hindering comparison and reproduction.

3 Empirical Analysis: Setup

We have shown that there exist various approaches to extracting TTPs from CTI reports. However, due to the wide variety of approaches, datasets, ontologies onto which TTPs are mapped and evaluation scenarios, it remains unclear to what extent methods leveraged by these approaches affect the performance on TTP extraction. Therefore, in this section, we design an experimental setup that allows us to compare the different approaches and measure the influence of their individual methods. Given the variety in methods, and customization to specific ontologies and datasets, we do not aim to compare individual works, but instead provide a setup by which we can analyze and understand the individual methods, as well as their advantages and limitations with respect to TTP extraction from CTI reports. Therefore, we decided to not directly re-implement existing solutions but rather implement and evaluate their underlying core methods for extracting the TTPs. This allows us to evaluate how suitable existing methods are for the task of TTP extraction without taking data-specific and ontology-specific optimizations into account. However, we must establish a framework by which we measure the performance of individual methods. This includes choosing an ontology of TTPs that allows us to evaluate whether TTPs are correctly extracted; a (labeled) dataset used for measuring the performance; and metrics that allow us to compare different methods and understand their advantages and limitations.

3.1 Ontology

There are many standards for threat actions by academics and industry. The STIX 2.1 [6] format has widely been adopted as a standard format by many organizations to describe cyber threat intelligence as it provides a generic and extendable framework. However, many other, more specific models and ontologies exist (often expressed in STIX 2.1), such as used by MISP [7] to express objects and galaxies and MITRE's ATT&CK [89], D3FEND [8], CAPEC [8], CWE [8] and CVE [8] frameworks. The ATT&CK framework focuses mainly on TTPs, common Threat Groups, and Software and forms the de facto standard for expressing TTPs. The ATT&CK framework consists of a matrix that maps procedures to techniques and techniques to corresponding tactics. These TTPs are based on real-world observations. Due to the older datasets in use (discussed later in Section 3.2), we follow ATT&CK Enterprise Version 14.1.

[6] https://oasis-open.github.io/cti-documentation/
[7] https://www.misp-project.org/
[8] https://{d3fend,capec,cwe,cve}.mitre.org/

Table 4: Comparison of MITRE ATT&CK labeled datasets

Dataset	Gran.	#CTRs	#Sent.	#Labels TA	T/ST	S	G
AnnoCTR [50]	Word	120	4,491	12	133	63	38
TRAM2 [82]	Sent.	150	19,011	×	50	×	×
CTI-to-MITRE [72]	Sent.	×	12,945	×	188	×	×
Kumarasinghe et al. [49]	Sent.	×	12,006	×	410	×	×
rcATT [52]	Doc.	1,490	×	12	215	×	×
MITREtrival [37]	Doc.	700	×	14	165	×	×

Legend: (Gran.) Granularity of labels. (T) Techniques. (ST) Sub-Techniques. (TA) Tactics. (G) Groups. (S) Software.

3.2 Datasets

The selection of public datasets containing cybersecurity reports labeled according to the ATT&CK framework is limited, as shown in Table 4. The largest and most known dataset comes from the MITRE corporation and has been released as part of a project focused on performing automatic TTP extraction, known as Threat Report ATT&CK Mapper (TRAM) [82]. The dataset contains original CTI reports with experts' annotations on individual sentences, and whole documents, allowing for the evaluation of approaches that operate on different granularity levels. However, the dataset contains only 50 (sub-)techniques from the ATT&CK framework that most frequently appear in real documents [82]. AnnoCTR [50] represents a similar effort, with annotation for more (sub-)techniques (133), and other information (e.g., adversary tactics, malware, threat actor groups), but its size is sensibly smaller. Other datasets, such as MITREtrival [37] and rcATT [52], are not suitable for sentence-based methodologies due to their exclusive document-level labeling, and automated or unvalidated labeling processes. CTI-to-MITRE [72] and Kumarasinghe et al. [49] have a large variety of unique (sub-)techniques but lack sentences without techniques, which is relevant for replicating a realistic evaluation scenario. Therefore, we select TRAM2 and AnnoCTR for our experimental validation, as they possess the best overall features. We divide the TRAM2 reports into a training and a test set, following the standard 80-20 split ratio. This results in 3,829 of the 19,178 technique labels being allocated to the test set. Entire reports are randomly assigned to either the training or test set to maintain compatibility with document-based approaches and evaluation metrics while ensuring that the context of each report remains intact. For AnnoCTR, we refer to the splits proposed by the authors [50]. We do not combine TRAM2 with datasets proposed by other works [49, 72], because they lack real CTI report sentences. While non-technique sentences can be manually extracted from MITREtrieval [37] or rcATT [52], adding them to these datasets breaks contextual integrity and affects context-based approaches. In conclusion, we rely solely on real-world CTI datasets to ensure consistency.

3.3 Metrics

Recognizing ATT&CK (sub-)techniques requires a multi-label classification setting, whether at the document or sentence level, as multiple labels can apply simultaneously. Given the imbalanced class ratios in our experimental datasets (Section 3.2), accuracy is not a meaningful metric. Instead, we use Precision, Recall, and the F1-Score, which provide a more accurate representation of performance. Precision measures the correctness of predictions, while Recall indicates the proportion of correct labels found, and the F1-Score represents the harmonic mean of these two metrics. Let P be the set of predicted labels and T the set of true labels. Precision is defined as $Precision = |T \cap P|/|P|$, while Recall is defined as $Recall = |T \cap P|/|T|$. The harmonic mean, the F1-Score, can then be defined as follows: $F_1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$. We calculate the F1-Score per document. For sentence-based models, sentence-level predictions are combined and compared to the ground truth, while document-based models provide predictions directly at the document level. The final F1-Score is obtained by averaging document-level scores across all documents. Note that the average F1 score cannot be calculated from the average Recall and Precision.

4 Empirical Analysis: NER

In this section, we evaluate to what extent the aforementioned NLP pipeline is able to detect TTPs in CTI reports and study the contribution of the individual pipeline components.We discuss each component, the use case-specific adaptations to identify TTPs, and their influence on the detection process.

4.1 Methods and Adaptations

We recall that NER approaches apply several techniques, often in a pipeline, to process raw text when performing NER. However, the individual components can be adapted to domain-specific use cases. Here, we discuss potential adaptations for extracting TTPs from CTI reports.

Tokenization. Tokenization splits strings of text into words and sentences that allow NER to identify words as entities. Traditionally, tokenization uses a preset vocabulary combined with rules and exceptions to deal with punctuation, affixes, and word boundaries. While CTI reports are often written in natural English language, they contain parts such as IP addresses, hashes or other Indicators of Compromise (IoCs) that are not necessarily covered by generic methods but would be useful to be recognized as individual tokens in further processing. Therefore, we adapt the tokenizer to detect IoCs using regular expressions (available in our repository).

Part-of-Speech Tagging. Identifying part-of-speech (POS) tags allows NER to distinguish e.g., actions (verbs) from entities (nouns). These tags are often assigned using a combination of rules (closed class POS tags) and machine learning models (open class tags). While POS tagging generally does not require much adaptation, we include a list of exceptions based on the ATT&CK framework to correct for special use cases[9], such as the scheduling software "at" which would be detected as an adposition, but should be recognized as a noun. However, Section 4.2 shows that POS tagging has limited influence on the performance of NER approaches.

Lemmatization. Lemmatization reduces tokens to a base form to ensure that different forms of words can be identified by NER (e.g., identifying “attacking” when looking for “attack”). In this work, we pay special attention to IoCs, which in CTI reports are often defanged (e.g., an IP address is written as 127[.]0.0.1) to ensure these do not accidentally trigger anti-virus systems or are accessed by humans. Instead, during lemmatization we “refang” these IoCs to ensure different defang methods are considered equivalent during NER.

Related Word Detection. Besides different forms of words, NER should also recognize words with the same intent (e.g., when recognizing “executing code”, “running” is a relevant synonym of executing; “killing”, however, is not). There are two main approaches to identifying word relationships: using knowledge bases of manually identified relations and using text embeddings. We focus on the creation of a knowledge base as text embeddings used and evaluated for classification-based approaches in Section 5.2. Therefore, we adapt WordNet [25], a database capturing semantic relations between terms, by limiting it to terms present in the ATT&CK framework and all the synonyms of these terms manually identified to be relevant to the cybersecurity domain. We identified 3,200 unique terms in the ATT&CK framework, of which 1,237 contained one or more synonyms in WordNet. For overlapping terms, we found a total of 2,369 matching synonym sets that include a total of 1,428 unique terms in 5,465 synonym relations. These synonyms can be used to broaden the search for named entities, which will be evaluated in Section 4.2.

Parsing. The parsing component identifies the semantic relations between words within sentences, e.g. the subject and object of a sentence. While there exist domain-specific approaches for parsing, such as Semantic Role Labeling (SRL) [33] to attach domain-specific roles to parts of a sentence, this is not required for NER. Therefore, in this work, we only evaluate generic parsing methods.

Named Entity Recognition. Using the previous components and their adaptations, we can perform NER by defining target classes (in our case, ATT&CK techniques) and searching for explicit references to those classes in the CTI reports. Here we make a distinction between two cases: single-token entities and subphrase matching. For single-token entities we simply match on the lemma or related word of the known entity. In the case of subphrase matching, we can either perform an exact match, e.g., find the term “Privilege Escalation“ or leverage parsing to detect different sentence orders (e.g., “escalating

[9]The full list of exceptions is available in our repository

privileges") or even ignore irrelevant intermediate tokens such as "The privileges of the attacker were escalated". We leverage parse trees to only detect direct relations between terms to prevent detecting false positives such as "Command and Control" which could otherwise be detected in the sentence "The user controlled their program through the command-line interface." Section 4.2 discusses the difference in performance for these approaches.

4.2 Evaluation

We perform an ablation study to determine the contribution of each pipeline component to the overall NER performance. Here, we disable one of the components and compare the drop in performance compared to using the full pipeline. We evaluate the performance of NER using the TRAM2 dataset as discussed in Section 3.2. While the NER approach does not require any training data (and therefore could theoretically use the full dataset for evaluation), we limit the evaluation to the test part of the dataset used in Sections 5.2 and 6.2 to ensure a fair comparison between the different approaches.

Ablation Study. Table 5 shows the results of the ablation study in which we compare the full pipeline with both a base pipeline that performs exact matching, as well as pipelines with a single component disabled. We find that exact matching is the most precise, i.e. will more rarely find techniques that are not present, due to their explicit mentioning. However, this also results in a lower recall compared to the other approaches, where we find that components sacrifice precision for an increase in recall. This is especially notable in the lemmatization and parsing components which improve matching for individual tokens and subphrases respectively. Finally, we find that disabling the related word component can actually improve performance in terms of both precision and F1-score, meaning that synonyms may not always be relevant when detecting named entities. In short, individual components especially contribute to improving the recall of NER approaches at the cost of false detections.

Table 5: Ablation study for NER pipeline.

	AnnoCTR (%)			TRAM2 (%)		
Approach	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Full pipeline	43.50	70.96	53.94	65.42	**57.83**	61.39
Base pipeline	47.26	66.25	55.17	**70.50**	47.29	56.61
No POS	40.11	**71.01**	51.26	65.22	56.60	60.61
No Lemmatization	**53.88**	66.90	**59.69**	69.23	48.45	57.00
No Related Words	45.46	68.99	54.81	67.69	57.51	**62.19**
No Parsing	40.28	64.92	49.71	65.36	49.16	56.12

5 Empirical Analysis: Classification

In this section, we compare and discuss the performance of the two main text classification methods for the extraction of TTPs presented in Section 2.3.

5.1 Methods and Adaptation

As discussed in Section 2.3, text classification approaches rely on different methods. The following sections present the adaption of state-of-the-art approaches.

Embedder model. We retrieve the main publicly available BERT-based models that have been adopted in CTI literature and subdivide them into categories that reflect the nature of the dataset they have been pre-trained on. We also include a model among the highest-ranking models for semantic text similarity according to the MTEB leaderboard [69]. The full list of models is available in Appendix D.

Labeled text classification. We replicate the common labeled text classification approach proposed in Table 2 [6, 56, 72, 77, 79, 80, 98, 100]. Each pre-trained BERT model is combined with a classification head, a dense layer with neurons matching the number of labels (50 for TRAM2 and 118 for AnnoCTR), and a sigmoid activation function to output log probabilities. We fine-tune each model on the training set with a learning rate of $2 * 10^{-5}$ and a batch size of 16. These values are concordant with the ones used in other literature works [6, 72], and with those proposed by studies on this topic [90]. To address label imbalance, we also evaluate the inclusion of a weight for positive labels in the loss function. For fine-tuning, we use 20% of the training data for validation and train for up to 100 epochs with early stopping and a patience parameter to prevent overfitting.

Data augmentation. We replicate two data augmentation strategies: one based on the inclusion of artificial data and the other on Out-of-Distribution (OOD) data. Note that many synthetic data generation strategies exist, but we focus on text rephrasing with generative LLMs, which have gained traction in NLP research [17]. For a more in-depth analysis of these strategies, we refer to related survey efforts [53, 108]. For the first strategy, we use the approach of Hao et al. [35] to generate synthetic variations of the TRAM2 dataset using the LLama-3.3_{70B} model. We then select sentences with cosine similarity between 0.3 and 0.9[10] to the original samples using SBERT (MPNet), and include them while preserving the original label distribution. For the second strategy, we follow You et al. [100] by including procedure descriptions from the ATT&CK framework with corresponding labels as OOD data. The resulting training sets are 63.06% and 22.24% larger than the original TRAM2 training set.

[10] Hao et al. [35] use 0.3 and 0.8, however, we observed that with 0.8 the size of the augmented dataset did not increase by a sufficient margin.

Table 6: Test results of labeled approaches. Please note that since TRAM2's model has been already trained on a different split of the TRAM dataset, on such dataset, we test the model as provided by the authors.

		AnnoCTR (%)			TRAM2 (%)		
Model	Dom.	F1	Prec.	Rec.	F1	Prec.	Rec.
CyBERT [78]	CTI	52.28	50.51	58.28	62.72	52.09	87.16
CySecBERT [10]	CTI	**62.75**	60.75	**67.86**	69.74	79.35	66.76
DarkBERT [41]	CTI	54.90	68.73	49.51	66.67	78.95	63.55
SecBERT [60]	CTI	55.01	59.39	54.80	63.67	53.28	83.86
SecRoBERTa [60]	CTI	50.28	58.17	46.65	59.86	47.88	87.73
SecureBERT [2]	CTI	58.44	70.94	53.13	70.24	74.55	70.67
TRAM2 (SciBERT) [82]	CTI	52.69	73.11	44.87	66.64	82.38	57.51
$BERT_{Base, Cased}$ [22]	Gen.	50.24	73.58	43.21	66.55	**82.56**	60.33
$BERT_{Base, Uncased}$ [22]	Gen.	52.77	49.72	60.15	63.69	53.44	86.64
$RoBERTa_{Base}$ [62]	Gen.	50.00	70.56	41.57	62.05	77.61	54.86
$RoBERTa_{Large}$ [62]	Gen.	54.26	66.03	50.31	**70.55**	77.59	68.39
$XLM\text{-}RoBERTa_{Base}$ [62]	Gen.	53.58	66.58	48.08	57.13	43.82	**92.70**
$XLM\text{-}RoBERTa_{Large}$ [62]	Gen.	57.32	**75.41**	48.90	61.93	81.73	53.42
$SciBERT_{Cased}$ [11]	Sci.	56.67	74.87	49.22	61.61	51.08	83.28
$SciBERT_{Uncased}$ [11]	Sci.	59.07	62.38	59.94	62.20	52.89	82.07

Table 7: Test results of unlabeled approaches.

		AnnoCTR (%)			TRAM2 (%)		
Model	Dom.	F1	Prec.	Rec.	F1	Prec.	Rec.
ATTACK-BERT [1]	CTI	**35.45**	**35.72**	40.73	**50.60**	**43.07**	**67.95**
SentSecBERT [49]	CTI	31.74	30.53	47.25	36.71	35.71	52.20
NV-Embed-V2 [51]	Gen.	29.51	24.17	**62.45**	45.69	41.59	64.94
SBERT (MPNet) [81]	Gen.	34.58	33.64	46.49	44.58	41.00	55.59
		AnnoCTR (All TTPs)			TRAM2 (All TTPs)		
Model	Domain	F1	Prec.	Rec.	F1	Prec.	Rec.
ATTACK-BERT [1]	CTI	**13.44**	**8.93**	41.46	10.71	7.24	**67.95**
SentSecBERT [49]	CTI	10.45	6.18	47.49	10.32	7.26	52.20
NV-Embed-V2 [51]	Gen.	9.50	5.85	**63.19**	11.37	7.85	64.94
SBERT (MPNet) [81]	Gen.	12.85	8.23	48.00	**12.23**	**8.31**	55.59

Unlabeled text classification. We adapt unlabeled text classifiers following an approach inspired by Alam et al. [5]. Using sentence embedding models (Appendix D), we create embeddings for ATT&CK TTP titles and descriptions, concatenating them into a single string. We then calculate the similarity between these embeddings and those of input sentences using the same model. Similarity values above a threshold lead to the class assignment, with the optimal threshold selected by minimizing the average F1-score on the validation set.

5.2 Evaluation

Comparison of labeled and unlabeled approaches. In this experiment, we compare the performances of labeled and unlabeled approaches on our two experimental datasets (see Section 3.2). As shown in Tables 6 and 7, labeled approaches outperform unlabeled ones, with RoBERTa achieving the highest F1-Score of 70.55% on TRAM2 and CySecBERT scoring 62.75% on AnnoCTR. In contrast, unlabeled approaches reach F1-Scores of 50.60% and 35.45%. Both labeled and unlabeled approaches achieve Recall values above 60%, but unlabeled models score generally lower Precision values, likely indicating that the models are providing few correct labels with their answers. Interestingly, the difference in highest F1-Score values between the two datasets is smaller for labeled approaches (7.8%) than for unlabeled ones (15.15%). Consider that AnnoCTR has 68 additional labels than TRAM2, and a significantly lower number of samples, hinting that labeled approaches are less impacted by the increase of labels. Hyper-parameter tuning has a relatively small impact on $RoBERTa_{Large}$'s performance since we were able to push its F1-Score by 0.11% (see Appendix C for more details). As shown in Table 7, unlabeled approaches can also output any possible (sub-)technique similar to the input sentence, but all the models undergo a significant drop in all evaluation metrics, and appear to be practically unusable due to their low Precision (see Table 7). Lastly, both Tables also show that models pre-trained on CTI do not show a consistent improvement on generic models, and even on models that have been pre-trained on scientific language, such as SciBERT [11]. The highest performances on TRAM2 are achieved by a general language model, while on AnnoCTR by a CTI-specific model, CySecBERT. For the unlabeled approaches, instead, ATTACK-BERT outperforms other models on both datasets, with a larger margin on AnnoCTR, around 5%.

Table 8: Test results of data augmentation.

	TRAM2		
Augmentation	F1 (%)	Prec. (%)	Rec. (%)
Synthetic Data	65.42 (-5.13)	71.96 (-5.63)	63.12 (-5.27)
OOD Data	**71.41 (+0.86)**	**72.76 (-4.83)**	**74.53 (+6.14)**

Data augmentation. Recall that data augmentation is a component exclusive to labeled approaches. In this experiment, we fine-tune the $RoBERTa_{Large}$ model, which achieves the highest test score on the TRAM2 dataset (see Table 6), on two augmented versions of the original TRAM2 dataset—the augmentation process is described in Section 5.1. The results in Table 8 suggest that both data augmentation approaches decrease the Precision of our reference model. However, the augmentation based on the inclusion of Out-of-Distribution data [100] can provide slightly beneficial results, with an F1-Score increase of 0.86% on the non-augmented model.

6 Empirical Analysis: Generation

This section presents the implementation and evaluation of the automatic TTP extraction methods reviewed in Section 2.4.

6.1 Methods and Adaptation

The selection of generative LLMs is immense and almost daily new SotA models are released in different parameter sizes or fine-tuned derivatives that lead to new high scores

on leaderboards such as Open LLM Leaderboard V2 [11] or Chatbot Arena [16]. However, for the sake of our systematisation of knowledge, we focus on the Llama3.1-Instruct$_{8B}$ model [63].[12] It is popularly used in research, is open source and has feasible dimensions allowing for reasonable reproducibility. We use the transformers library from huggingface [13] with unsloth [34] as the fine-tuning library. Low-Rank Adaptation (LoRA) is used as the training method on all target modules [36]. Fine-tuning is always performed with a constant learning rate of $1e-5$ for the embedding layer and $2e-5$ for all other layers, with a batch size of 4 for 3 epochs. The embedding model used for our generative LLM RAG experiments is Qwen2-Instruct$_{7B}$ [57], which ranked first for English texts in the Massive Text Embedding Benchmark [69] in July 2024. Quantization is omitted and all models are trained and evaluated on 16-bit weights. All tests are repeated 3 times on different seeds on a single NVIDIA L40.

Response Interpretation. Generative LLMs classify by generating text, which must be interpreted to extract classes. We explore two approaches: matching MITRE concepts by their names and matching them by their IDs. Both methods yield similar results in our experiments. For the sake of clarity, we therefore only show the ID extraction approach in all following experiments. Table 14 in Appendix B shows the minimal differences of both approaches.

6.2 Evaluation

Overall, all methods can be divided into two categories: prompt-based and weight-based. Prompt-based methods change the prompt to give the LLM more information during inference time, such as Few-Shot Prompting (FSP) and Retrieval Augmented Generation (RAG). Weight-based methods instead change the weights during the training phase, i.e. mainly Supervised Fine-tuning (SFT). All results are summarized in Table 9, where *Raw* indicates a static prompt, using suggested prompt-engineering techniques [83], with no additional methods.

Prompt-based. FSP and RAG are methods that are both realized directly on the prompt. One behavior we have observed is the instability of the prompt. The language model can be unstable and produce irrelevant results when given slightly modified prompts, leading to hallucinations and decreased accuracy in extracting relevant information. Prompts can be overly unstructured, unclear, or simply inappropriate for the model causing F1-scores of < 0.10 on both tested datasets. Meanwhile our *Raw* prompt achieves an F1-score of 0.30 on the AnnoCTR test dataset and 0.49 on the TRAM2 test dataset. This instability of prompts is also observed in the work of Fieblinger et al. [26] during TTP extraction.

Table 9: Comparison results of Generative LLM methods.

Category	Method	AnnoCTR (%)			TRAM2 (%)		
		F1	Prec.	Rec.	F1	Prec.	Rec.
Prompt-Based	Raw	30.3	24.4	50.8	49.2	39.6	65.1
	FSP	26.4	34.1	24.5	46.2	47.5	44.9
	RAG	35.9	32.7	53.1	54.1	45.5	66.8
	RAG + FSP	34.6	33.3	39.3	53.2	48.5	65.5
Weight-Based	SFT Raw	**55.3**	**50.8**	**60.7**	**72.5**	**66.3**	**80.0**
Weight + Prompt	SFT FSP	42.8	53.3	39.8	57.9	51.3	69.5
	SFT RAG	47.6	48.7	54.9	65.5	57.6	76.0
	SFT RAG + FSP	47.3	49.2	54.0	64.6	56.0	76.4

Few-Shot Prompting. We also observed similar behavior with FSP. We intentionally did not adapt the examples to the test data records, as this is not an option in a real-world scenario. For this reason, we randomly selected 5 example sentences with the correct labels and added them to the prompt. Without precise adjustments to the existing test dataset, such as using labels that appear more often, the result is worse than without FSP. This behavior of FSP is a known phenomenon explained in many papers how badly chosen examples can cause a bias and degrade performance [15, 26, 49, 107].

Retrieval Augmented Generation. RAG has the advantage that it uses a semantic similarity check to add only (supposedly) helpful information to the prompt. We compared the semantic embeddings from the sentence under investigation with embeddings that we generated from all ATT&CK concept descriptions and names in order to present possible ATT&CK concept candidates in the prompt. We achieve the best results by adding five RAG entries to the prompt. In Table 9 we see that mainly the precision of the model improves, as it can probably make more confident decisions with the additional RAG information names, IDs and corresponding descriptions.

Weight-based. The advantage of SFT is that it is more independent of the prompt, as the model is taught what is to be derived from the prompt during training.

Supervised Fine-tuning. For training, we created LLM instruction data sets from the TRAM2 and AnnoCTR training dataset. As user input we used our normal TTP extraction prompt without any other methods. The output of the LLM consists of the corresponding label wrapped in 24 different templates created from the original responses of the untrained model. This has the advantage that the LLM does not have to change its general behavior, but is only trained on the correct tokens in the form of the correct IDs and names. The creation of the data set is not entirely trivial and can also lead to negative results. We were able to observe this negative behavior when the response of the LLM in the training dataset is simply a pure list of IDs. We also applied an idea from Zhang et al. [104] by creating the same training instruction dataset, with RAG entries, so that the model can learn to deal better

[11] https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

[12] Our study shows similar results for larger models with the same prompt, such as GPT-4o resp. the comparable (https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct) Llama3.3-70B-Instruct (Appendix B).

[13] https://huggingface.co/docs/transformers/index

Table 10: Test results of data augmentation for LLMs in comparison with original TRAM2 training set in parenthesis.

TRAM2			
Method	F1 (%)	Prec. (%)	Rec. (%)
Raw	**71.8** (-0.7)	70.7 (-9.3)	**72.9** (+6.6)
RAG	68.0 (+2.5)	78.0 (+2.0)	60.3 (-6.0)
FSP+RAG	69.9 (+5,3)	**81.0** (+4,6)	61.5 (+5,5)

Table 11: Document-level granularity for LLM strategies in comparison with sentence-level granularity in parenthesis.

AnnoCTR			
Method	F1 (%)	Prec. (%)	Rec. (%)
Raw	26.7 (-3.6)	31.4 (+7.0)	**26.2** (-24.6)
FSP	26.0 (-0.4)	27.3 (-6.8)	24.8 (+0.3)
RAG	**27.4** (-8.4)	**36.0** (+3.3)	25.8 (-27.3)
TRAM2			
Raw	**34.2** (-15.0)	**51.1** (+11.5)	**32.8** (-32.3)
FSP	31.0 (-15.2)	41.9 (-5.6)	30.9 (-14.0)
RAG	30.9 (-23.2)	45.8 (+0.3)	29.6 (-37.2)

with external context during training. With RAG applied, the results are not better than the fine-tuning without RAG entries. They are, however, more balanced in terms of recall and precision; detailed results are in Appendix B, Table 15.

Augmented Data. For the generative LLMs, we took the data set described in Section 5.1 and used it to create an LLM instruction data set with eight variations for each sentence. The distribution of the labels remains the same as in the original TRAM2 dataset. Although we trained with 8 times the amount of data, the results are close to the training on the original TRAM2 dataset, see Table 10. This demonstrates that data quality is crucial and does not rely on the quantity of semantically similar sentences.

Document-Level. Generative LLMs can also analyze entire CTI reports at once and provide estimates of the TTPs contained in one shot. The advantage is a faster analysis of the CTI reports and therefore fewer resources. As can be seen from Table 11, the precision increases slightly, while the recall for Raw and RAG drops by half in some cases across both datasets tested. RAG is more difficult at document-level because a similarity check over the entire content at once cannot achieve a short distance to all containing ATT&CK concepts. Even when splitting text into 500 characters-long passages, and adding multiple RAG for every text passage, this approach does not achieve results as good as sentence-level approaches.

7 Empirical Analysis: Overall Comparison

In this experiment, we compare the three approaches in a *closed-set scenario*, in which information on the data is known in advance, and an *open-set scenario*, where no prior information about the data set is available and no specific adjustments to the approaches is possible.

Table 12: Best performances on closed-set classifications of all three approaches on TRAM2 test dataset.

TRAM2			
Approach	F1 (%)	Prec. (%)	Rec. (%)
NER	62.19	57.51	67.69
Classification	71.41	72.76	**74.53**
Generation	**72.50**	**80.00**	66.30

Closed-set Scenario. This scenario is the most common in literature, as it allows the evaluation of methods that maximize the performance of the models: boundary conditions are already known in advance and can be used to gain advantage. For this test, we refer to the TRAM2 dataset, as it is the most widely used benchmark dataset. The results in Table 12 demonstrate that both Classification and Generation models outperform traditional NER approaches across all metrics, with Generation achieving the highest F1-Score of 72.50%, surpassing Classification by a margin of +1.09%.

Open-set Scenario. With the open-set scenario, we replicate a real-world setting in which the lack of data affects the capability to maximize the performances of our models. For this test, we adopt the AnnoCTR dataset, with label subsets of increasing frequency, namely the top-{10,25,50}, all AnnoCTR labels (118), and the entire ATT&CK Enterprise matrix (637). By having all the possible labels, we attempt to gain insights on a real-world, unconstrained scenario, assuming that every (sub-)technique can be referenced from the CTI report. Since NER is not a data-driven approach, the normal pipeline can be used here. For Classification, we refer to the unlabeled approach, which does not require training, with a classification threshold set to the reference value of 0.5. For Generation, we use a non-fine-tuned generative LLM with RAG entries taken from all the TTP descriptions. Figure 4 highlights that both Generation and Classification are inferior on every subset of labels, conversely to the closed-set scenario.

8 Limitations

Our analysis of methods for TTP extraction is subject to some limitations. First, no publicly available datasets fully satisfy the criteria outlined in Section 3.2, with the exception of TRAM2 [82], which we employ in this work. While TRAM2 encompasses the 50 most frequently observed techniques, its annotations remain constrained in scope. To address this limitation, we leverage AnnoCTR [50] (which covers 133 techniques), though its labels may still exhibit some degree of noise. Second, our study does not identify a definitive best-performing approach among the state-of-the-art methods explored. We focus on capturing the fundamental charac-

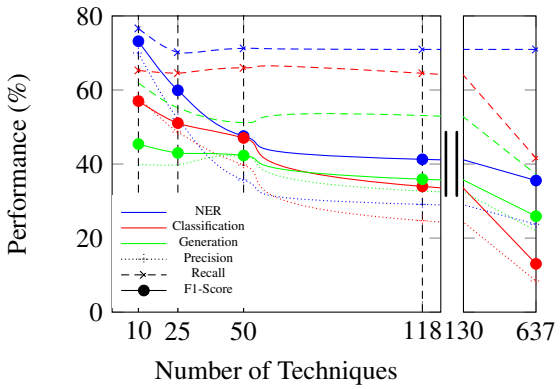

Figure 4: Comparison of performance on the top-k most common ATT&CK techniques for the AnnoCTR dataset in a (favorable) open-set-scenario.

teristics of such approaches, thereby providing actionable insights into the strengths and weaknesses of different NLP approaches and methods. Consequently, our implementations may not incorporate certain approach-specific optimizations (see Appendix A). Additionally, certain studies [5, 37, 49] evaluate combinations of multiple approaches, whereas our analysis focuses on individual methods. We believe that studying pipelined approaches is of interest for future research. Lastly, our research work focuses on the most recent public generative models. Given the rapid developments in generative AI, future architectures might yield better performance than those observed in this study.

9 Conclusion and Takeaways

Our results reveal that even the most recent NLP models, such as generative LLMs, are unable to surpass the F1-Score bar set at 70% by less recent models (e.g., BERT-based text classifiers), by more than a small degree. This phenomenon is consistent across both experimental datasets. Even though our experiments show that specific components aimed at solving challenges related to the application of NLP on cybersecurity language can provide additional support, the improvement is relatively small. We derive the following takeaways:

◇ **Lack of datasets.** Data-driven TTP extraction approaches overall achieve higher performances, but are fundamentally limited by the lack of manually annotated datasets. Many research works highlight this as the primary obstacle for building TTP extraction approaches. To mitigate this issue, some techniques, such as data augmentation, have been proposed to supplement limited datasets with synthetic samples, procedure descriptions extracted from knowledge bases like ATT&CK, or large-scale unsupervised pre-training of LLMs on cybersecurity texts. However, our results demonstrate that these methods are insufficient to compensate for the lack of high-quality, manually annotated data. Existing datasets show common limitations, such as the inclusion of a subset of TTPs (e.g., TRAM2 only covers 50 techniques), and strong imbalance among the included labels. In addition, the building process of some of the proposed datasets lacks sufficient clarity. Overall, dataset-related issues also affect the benchmarking process of NLP techniques, making it harder to understand their strengths and weaknesses. The research and industry communities should focus on providing higher quality datasets, including annotations for entire original documents, enabling the evaluation of the downstream task.

◇ **Label confusion.** An in-depth analysis of our results reveals that treating TTP extraction as a single-sentence classification problem may be inherently limited. The outputs from our best-performing Text Classification model, $\text{RoBERTa}_{\text{Large}}$, show that certain labels are often confused or appear together incorrectly in the answers of the model. For instance, techniques like T1112 and T1547.001, or T1140 and T1027, are frequently misclassified (see Appendix F for more information). Sometimes, this confusion is due to subtle differences in meaning, such as the distinction between file obfuscation (T1207) and deobfuscation (T1140). Most likely, the embeddings of sentences describing this behavior are close enough to confuse the classifier, and more advanced NLP techniques should be able to circumvent such issues. In other cases, we can instead derive that their differences are likely too subtle to be captured by analyzing single sentences, even for a human expert. For example, both T1547.001 and T1112 involve modifying the Windows Key Registry, but the context and intentions behind these actions differ significantly. T1547.001 is related to achieving persistence for autostart, while T1112 is associated with hiding information or aiding in persistence. For example, in TRAM2, T1112 is associated with the sentence *"If it is not installed, it installs itself to %programdata% and then sets the registry run key for persistence."*, while T1547.001 to *"It sets up a run key for via the command C:\ProgramData\GoogleUpdate\googleupdate.exe work for persistence."*. Previous context information and reasoning over the attacker's motives are key to distinguishing these two scenarios. In AnnoCTR [50], an agreement test shows that two different annotators only have a 31% agreement on the ATT&CK concepts. All these observations lead us to suspect that the ATT&CK framework is not always sufficiently unambiguous. This suspicion should be investigated further to see to what extent it also represents a potential source of error. We also derive the following: first, future automated extraction tools need to incorporate additional context beyond single sentences to accurately capture the intended goals of threat actions; second, while existing machine-supported annotation tools [1, 49] represent a step in the right direction, future works should focus on providing expert annotators with more context to ensure a correct labeling process.

◇ **Approach Summary.** In this work, we have subdivided and analyzed existing works into three main categories, rule-based NER systems, data-driven Classification, and Generation sys-

tems, discussing their respective benefits and drawbacks. Due to their rule-based nature, NER systems provide explainable outputs but require careful adaptations to account for the peculiarities of cybersecurity language. Such systems also show reliable results with limited knowledge about data. When labeled data is available, classification models provide improvements over NER systems (see Section 7), and their performances are seemingly on par with generative models. We also observed that data augmentation can provide marginal benefits, while the adoption of embedding models pre-trained on CTI-domain corpora does so less consistently (see Tables 6, 7). As the popularity of BERT-based models has been obscured by more recent decoder-only transformers [94], future research should still investigate such methods due to their performance-size tradeoff. Supervised fine-tuned generative LLMs achieve comparable results to classification models. However, the creation of the datasets, the prompts, the response interpretation, the provision of the required computing power, and the evaluation of all methods is more difficult than with many BERT-like models of classification. If there is not enough qualitatively labeled data or sufficient machine learning expertise available, an untrained generative LLM can be used in some settings as an alternative to classification models, as can be seen in Figure 4. Nevertheless, it must be noted that generative LLMs are not the best tool for every task.

◇ **Recommendations.** Practitioners seeking to include a TTP extraction process in their CTI pipeline must audit their annotated data availability, identify which subset of ATT&CK's patterns are of interest, and guide the model development accordingly. Generative LLMs are tempting: they may solve this task under conditions where training data is completely absent and allow for the development of approximate systems simply with a natural language description of the task. However, results obtained with this approach are often error-prone. More consistent results can be obtained with traditional NLP methods (although complex to maintain and develop), or with smaller and less resource-hungry models (e.g., BERT) if enough training data is provided. Modern threat hunting, and consequently, strategic decisions are guided by the analysis of TTPs [18]. Systems with low precision (e.g., noisy malware behavior detection via SIGMA rules) risk misleading analysts with false positives, while low recall systems omit critical adversary capabilities, creating incomplete threat profiles. High-recall NER systems aid in extracting relevant entities from incident reports in real-world, alarm-overloaded environments but require analyst filtering due to low precision, making them better suited as data-labeling recommendation tools. Conversely, Classification approaches – trained on high quality datasets – already strike a good balance for closed TTP sets. Generative approaches offer triage assistance but introduce hallucination risks. Recent "Large Reasoning Models" seem to be subject to this issue as well [93]. Both options – maintaining traditional NLP pipelines or monitoring LLM outputs – carry costs that depend on the team's capacity. NLP methodologies and their optimizations must be carefully evaluated on real-world CTI data to prevent overfitting.

◇ ***Are We There Yet?*** Our results suggest that TTP extraction cannot yet realistically cover the entire ATT&CK framework. Even with prior knowledge of the test dataset, the results of all experiments are far from being fully reliable automatic annotation systems for practical use. While there are technical advances in the individual approaches, our reasonable suspicion is that work on automatic TTP extraction is being held back not by a lack of innovation in NLP-based applications, but by a lack of high quality datasets. Existing datasets are small, offer only a subset of existing labels, suffer from class imbalance, and may contain ambiguous annotations. Besides, the research community focuses almost exclusively on ML-based approaches in unrealistic closed-set scenarios, without experiments in more realistic open-set scenarios.

Ethics Considerations

This research systematizes the knowledge on the automated extraction of TTPs. By providing a comprehensive analysis of relevant methodologies, it aims to advance defensive cybersecurity and threat analysis, rather than offensive applications. Specifically, automating repetitive, manual-intensive extraction tasks enhances the work of cybersecurity professionals. The study follows ethical, replicable, and fair research practices, using public datasets, established frameworks (e.g., MITRE ATT&CK), rigorous validation, and responsible model sharing. We acknowledge **stakeholders**–cybersecurity professionals relying on accurate TTP extraction, organizations needing timely threat insights, researchers, and the public benefiting from stronger defenses–as well as **risks**, including tool misuse, model hallucinations, biases, and data sensitivity. Misuse is **mitigated** through responsible open-source sharing, a defined defensive scope, and security guidelines. Bias and hallucination risks are reduced through validation against the MITRE ATT&CK framework, carefully crafted prompts, and diverse training data. Sensitive data risks are addressed by using public sources and adhering to established security and privacy protocols.

Open Science

We are committed to the open science policy and made all our source code, models and datasets publicly available at https://doi.org/10.5281/zenodo.15608555.

Acknowledgments

We would like to thank our reviewers for their valuable inputs. Tommaso Paladini acknowledges support from TIM

S.p.A. through the Ph.D. scholarship. This work is also supported by the SeReNity project, Grant No. cs.010, funded by Netherlands Organisation for Scientific Research (NWO).

References

[1] Basel Abdeen, Ehab Al-Shaer, Anoop Singhal, Latifur Khan, and Kevin W. Hamlen. SMET: semantic mapping of CVE to att&ck and its application to cybersecurity. In Data and Applications Security and Privacy XXXVII - 37th Annual IFIP WG 11.3 Conference, DBSec, Lecture Notes in Computer Science. Springer, 2023.

[2] Ehsan Aghaei, Xi Niu, Waseem G. Shadid, and Ehab Al-Shaer. Securebert: A domain-specific language model for cybersecurity. In Security and Privacy in Communication Networks - 18th EAI International Conference, SecureComm, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2022.

[3] Kashan Ahmed, Syed Khaldoon Khurshid, and Sadaf Hina. Cyberentrel: Joint extraction of cyber entities and relations using deep learning. In Computers & Security, 2024.

[4] Bader Al-Sada, Alireza Sadighian, and Gabriele Oligeri. MITRE att&ck: State of the art and way forward. In ACM Computing Surveys, 2025.

[5] Md Tanvirul Alam, Dipkamal Bhusal, Youngja Park, and Nidhi Rastogi. Looking beyond iocs: Automatically extracting attack patterns from external CTI. In Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses, RAID. ACM, 2023.

[6] Paulo M. M. R. Alves, Geraldo P. R. Filho, and Vinícius P. Gonçalves. Leveraging bert's power to classify ttp from unstructured text. In 2022 Workshop on Communication Networks and Power Systems (WCNPS), 2022.

[7] Gbadebo Ayoade, Swarup Chandra, Latifur Khan, Kevin W. Hamlen, and Bhavani Thuraisingham. Automated threat report classification over multi-source data. In 4th IEEE International Conference on Collaboration and Internet Computing, CIC. IEEE Computer Society, 2018.

[8] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and lemmatization: A comparison of retrieval performances. Lecture Notes on Software Engineering, IACSIT Press, 2014.

[9] Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. UKP: computing semantic textual similarity by combining multiple content similarity measures. In Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT. The Association for Computer Linguistics, 2012.

[10] Markus Bayer, Philipp Kuehn, Ramin Shanehsaz, and Christian Reuter. Cysecbert: A domain-adapted language model for the cybersecurity domain. In ACM Transactions on Privacy and Security, 2024.

[11] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP. Association for Computational Linguistics, 2019.

[12] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching word vectors with subword information. In Trans. Assoc. Comput. Linguistics, 2017.

[13] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS, 2020.

[14] Minghao Chen, Kaijie Zhu, Bin Lu, Ding Li, Qingjun Yuan, and Yuefei Zhu. AECR: automatic attack technique intelligence extraction based on fine-tuned large language model. ACM Computers & Security, 2025.

[15] Yutong Cheng, Osama Bajaber, Saimon Amanuel Tsegai, Dawn Song, and Peng Gao. CTINEXUS: leveraging optimized LLM in-context learning for constructing cybersecurity knowledge graphs under data scarcity. CoRR, abs/2410.21060, 2024.

[16] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference. In Forty-first International Conference on Machine Learning, ICML, 2024.

[17] Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Fang Zeng, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. Auggpt: Leveraging chatgpt for text data augmentation. In IEEE Transactions on Big Data. IEEE, 2025.

[18] Roman Daszczyszak, Dan Ellis, Steve Luke, and Sean Whitley. Ttp-based hunting. MITRE Corp, McLean VA, Tech. Rep, 2019.

[19] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal stanford dependencies: A cross-linguistic typology. In Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC. European Language Resources Association (ELRA), 2014.

[20] Defense Advanced Research Projects Agency (DARPA). Darpa transparent computing program, 2014. https://www.darpa.mil/program/transparentcomputing.

[21] Isuf Deliu, Carl Leichter, and Katrin Franke. Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent dirichlet allocation. In IEEE International Conference on Big Data (BigData). IEEE, 2018.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805, 2018.

[23] Nuno Dionísio, Fernando Alves, Pedro Miguel Ferreira, and Alysson Bessani. Cyberthreat detection from twitter using deep neural networks. In International Joint Conference on Neural Networks, IJCNN. IEEE, 2019.

[24] Reza Fayyazi, Rozhina Taghdimi, and Shanchieh Jay Yang. Advancing TTP analysis: Harnessing the power of large language models with retrieval augmented generation. In Annual Computer Security Applications Conference, ACSAC. IEEE, 2024.

[25] Christiane Fellbaum. WordNet: An electronic lexical database. MIT press, 1998.

[26] Romy Fieblinger, Md Tanvirul Alam, and Nidhi Rastogi. Actionable cyber threat intelligence using knowledge graphs and large language models. In IEEE European Symposium on Security and Privacy Workshops, EuroS&PW. IEEE, 2024.

[27] Peng Gao, Xiaoyuan Liu, Edward Choi, Sibo Ma, Xinyu Yang, Zhengjie Ji, Zilin Zhang, and Dawn Song. Threatkg: A threat knowledge graph for automated open-source cyber threat intelligence gathering and management. CoRR, abs/2212.10388, 2022.

[28] Peng Gao, Fei Shao, Xiaoyuan Liu, Xusheng Xiao, Zheng Qin, Fengyuan Xu, Prateek Mittal, Sanjeev R Kulkarni, and Dawn Song. Enabling efficient cyber threat hunting with cyber threat intelligence. In 37th IEEE International Conference on Data Engineering, ICDE. IEEE, 2021.

[29] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. CoRR, abs/2312.10997, 2023.

[30] Wenhan Ge and Junfeng Wang. Seqmask: Behavior extraction over cyber threat intelligence via multi-instance learning. In The Computer Journal, 2024.

[31] Wenhan Ge, Junfeng Wang, Tongcan Lin, Binhui Tang, and Xiaohui Li. Explainable cyber threat behavior identification based on self-adversarial topic generation. Computer & Security, 2023.

[32] Yumna Ghazi, Zahid Anwar, Rafia Mumtaz, Shahzad Saleem, and Ali Tahir. A supervised machine learning based approach for automatically extracting high-level threat intelligence from unstructured sources. In 2018 International Conference on Frontiers of Information Technology, FIT. IEEE Computer Society, 2018.

[33] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. In Computer Linguistics, 2002.

[34] Daniel Han, Michael Han, and Unsloth Team. Unsloth, 2023.

[35] Zhiqiang Hao, Chuanyi Li, Xiao Fu, Bin Luo, and Xiaojiang Du. Leveraging hierarchies: HMCAT for efficiently mapping CTI to attack techniques. In Computer Security - ESORICS 2024 - 29th European Symposium on Research in Computer Security, Lecture Notes in Computer Science. Springer, 2024.

[36] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In The Tenth International Conference on Learning Representations, ICLR, 2022.

[37] Yi-Ting Huang, R. Vaitheeshwari, Meng Chang Chen, Ying-Dar Lin, Ren-Hung Hwang, Po-Ching Lin, Yuan-Cheng Lai, Eric Hsiao-Kuang Wu, Chung-Hsuan Chen, Zi-Jie Liao, and Chung-Kuan Chen. Mitretrieval: Retrieving MITRE techniques from unstructured threat reports by fusion of deep learning and ontology. In IEEE Transactions on Network and Service Management, 2024.

[38] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of CTI sources. In Proceedings of the 33rd Annual Computer Security Applications Conference. ACM, 2017.

[39] Ghaith Husari, Xi Niu, Bill Chu, and Ehab Al-Shaer. Using entropy and mutual information to extract threat actions from cyber threat intelligence. In 2018 IEEE International Conference on Intelligence and Security Informatics, ISI. IEEE, 2018.

[40] Hangyuan Ji, Jian Yang, Linzheng Chai, Chaoren Wei, Liqun Yang, Yunlong Duan, Yunli Wang, Tianzhen Sun, Hongcheng Guo, Tongliang Li, Changyu Ren, and Zhoujun Li. Sevenllm: Benchmarking, eliciting, and enhancing abilities of large language models in cyber threat intelligence. CoRR, abs/2405.03446, 2024.

[41] Youngjin Jin, Eugene Jang, Jian Cui, Jin-Woo Chung, Yongjae Lee, and Seungwon Shin. Darkbert: A language model for the dark side of the internet. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL. Association for Computational Linguistics, 2023.

[42] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. J. Documentation, 2004.

[43] Masashi Kadoguchi, Shota Hayashi, Masaki Hashimoto, and Akira Otsuka. Exploring the dark web for cyber threat intelligence using machine leaning. In 2019 IEEE International Conference on Intelligence and Security Informatics, ISI. IEEE, 2019.

[44] Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. Continual pre-training of language models. In The Eleventh International Conference on Learning Representations, ICLR, 2023.

[45] Heejung Kim and Hwankuk Kim. Comparative experiment on ttp classification with class imbalance using oversampling from cti dataset. Security and Communication Networks, 2022.

[46] Barbara A. Kitchenham and Pearl Brereton. A systematic review of systematic review process research in software engineering. Information and Software Technology, 2013.

[47] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS, 2022.

[48] Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. Dependency Parsing. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2009.

[49] Udesh Kumarasinghe, Ahmed Lekssays, Husrev Taha Sencar, Sabri Boughorbel, Charitha Elvitigala, and Preslav Nakov. Semantic ranking for automated adversarial technique annotation in security text. In Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS. ACM, 2024.

[50] Lukas Lange, Marc Müller, Ghazaleh Haratinezhad Torbati, Dragan Milchevski, Patrick Grau, Subhash Chandra Pujari, and Annemarie Friedrich. Annoctr: A dataset for detecting and linking entities, tactics, and techniques in cyber threat reports. In Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024. ELRA and ICCL, 2024.

[51] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. CoRR, abs/2405.17428, 2024.

[52] Valentine Legoy, Marco Caselli, Christin Seifert, and Andreas Peter. Automated retrieval of att&ck tactics and techniques for cyber threat reports. CoRR, abs/2004.14322, 2020.

[53] Bohan Li, Yutai Hou, and Wanxiang Che. Data augmentation approaches in natural language processing: A survey. AI Open, 2022.

[54] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. In IEEE Transactions on Knowledge and Data Engineering, 2022.

[55] Jingwen Li, Ru Zhang, and Jianyi Liu. Attack behavior extraction based on heterogeneous threat intelligence graphs and data augmentation. In International Joint Conference on Neural Networks, IJCNN. IEEE, 2024.

[56] Lingzi Li, Cheng Huang, and Junren Chen. Automated discovery and mapping att&ck tactics and techniques for unstructured cyber threat intelligence. In Computer & Security, 2024.

[57] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. CoRR, abs/2308.03281, 2023.

[58] Zhenyuan Li, Jun Zeng, Yan Chen, and Zhenkai Liang. Attackg: Constructing technique knowledge graph from cyber threat intelligence reports. In Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Lecture Notes in Computer Science. Springer, 2022.

[59] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem A. Beyah. Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

[60] Matteo Liberato. Secbert: Analyzing reports using bert-like models. Master's thesis, University of Twente, 2022.

[61] Chenjing Liu, Junfeng Wang, and Xiangru Chen. Threat intelligence att&ck extraction based on the attention transformer hierarchical recurrent neural network. In ACM Applied Soft Computing, 2022.

[62] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. CoRR, abs/1907.11692, 2019.

[63] Inc. Llama Team, AI @ Meta Platforms. The llama 3 herd of models. CoRR, abs/2407.21783, 2024.

[64] Christopher D. Manning. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing, Lecture Notes in Computer Science. Springer, 2011.

[65] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In 1st International Conference on Learning Representations, ICLR, 2013.

[66] Sadegh M. Milajerdi, Birhanu Eshete, Rigel Gjomemo, and V. N. Venkatakrishnan. POIROT: aligning attack behavior with kernel audit records for cyber threat hunting. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS. ACM, 2019.

[67] Mariella Mischinger, Sergio Pastrana, and Guillermo Suarez-Tangil. Ioc stalker: Early detection of indicators of compromise. In Annual Computer Security Applications Conference, ACSAC. IEEE, 2024.

[68] LLC MITRE Engenuity. Threat report att&ck mapper (tram). https://github.com/center-for-threat-informed-defense/tram, 2023.

[69] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: massive text embedding benchmark. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL. Association for Computational Linguistics, 2023.

[70] S Naveen, Rami Puzis, and Kumaresan Angappan. Deep learning for threat actor attribution from threat reports. In 2020 4th international conference on computer, communication and signal processing (ICCCSP). IEEE, 2020.

[71] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC. European Language Resources Association (ELRA), 2016.

[72] Vittorio Orbinato, Mariarosaria Barbaraci, Roberto Natella, and Domenico Cotroneo. Automatic mapping of unstructured cyber threat intelligence: An experimental study: (practical experience report). In IEEE 33rd International Symposium on Software Reliability Engineering, ISSRE, 2022.

[73] Tommaso Paladini, Lara Ferro, Mario Polino, Stefano Zanero, and Michele Carminati. You might have known it earlier: Analyzing the role of underground forums in threat intelligence. In The 27th International Symposium on Research in Attacks, Intrusions and Defenses, RAID. ACM, 2024.

[74] Youngja Park and Taesung Lee. Full-stack information extraction system for cybersecurity intelligence. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: EMNLP. Association for Computational Linguistics, 2022.

[75] Youngja Park and Weiqiu You. A pretrained language model for cyber threat intelligence. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP. Association for Computational Linguistics, 2023.

[76] Joël Plisson, Nada Lavrac, and Dunja Mladenic. A rule based approach to word lemmatization. In Proceedings of IS. Citeseer, 2004.

[77] Fariha Ishrat Rahman, Sadaf Md. Halim, Anoop Singhal, and Latifur Khan. ALERT: A framework for efficient extraction of attack techniques from cyber threat intelligence reports using active learning. In Data and Applications Security and Privacy XXXVIII - 38th Annual IFIP WG 11.3 Conference, DBSec, Lecture Notes in Computer Science. Springer, 2024.

[78] Priyanka Ranade, Aritran Piplai, Anupam Joshi, and Tim Finin. Cybert: Contextualized embeddings for the cybersecurity domain. In 2021 IEEE International Conference on Big Data (Big Data). IEEE, 2021.

[79] Nanda Rani, Bikash Saha, Vikas Maurya, and Sandeep Kumar Shukla. Ttphunter: Automated extraction of actionable intelligence as ttps from narrative threat reports. In Proceedings of the 2023 Australasian Computer Science Week, ACSW. ACM, 2023.

[80] Nanda Rani, Bikash Saha, Vikas Maurya, and Sandeep Kumar Shukla. Ttpxhunter: Actionable threat intelligence extraction as ttps from finished cyber threat reports. CoRR, abs/2403.03267, 2024.

[81] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP. Association for Computational Linguistics, 2019.

[82] James Ross and Jackie Lasky. Our tram large language model automates ttp identification in cti reports. https://medium.com/mitre-engenuity/our-tram-large-language-model-automates-ttp-identification-in-cti-reports-5bc0a30d4567, 2023.

[83] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. CoRR, abs/2402.07927, 2024.

[84] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR, abs/1910.01108, 2019.

[85] Kiavash Satvat, Rigel Gjomemo, and V. N. Venkatakrishnan. Extractor: Extracting attack behavior from threat reports. In IEEE European Symposium on Security and Privacy, EuroS&P. IEEE, 2021.

[86] Taneeya Satyapanich, Francis Ferraro, and Tim Finin. CASIE: extracting cybersecurity event information from text. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI. AAAI Press, 2020.

[87] Samaneh Shafee, Alysson Bessani, and Pedro M. Ferreira. Evaluation of llm-based chatbots for osint-based cyber threat awareness. Expert Syst. Appl., 2025.

[88] Giuseppe Siracusano, Davide Sanvito, Roberto Gonzalez, Manikantan Srinivasan, Sivakaman Kamatchi, Wataru Takahashi, Masaru Kawakita, Takahiro Kakumaru, and Roberto Bifulco. Time for action: Automated analysis of cyber threat intelligence in the wild. CoRR, abs/2307.10214, 2023.

[89] Blake E. Strom, Andy Applebaum, Doug P. Miller, Kathryn C. Nickels, Adam G. Pennington, and Cody B. Thomas. MITRE ATT&CK®: Design and Philosophy. In The MITRE Corporation, editor, MITRE, 2018. https://attack.mitre.org/, retrieved 2024-10-22.

[90] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? In Chinese Computational Linguistics - 18th China National Conference, CCL, Lecture Notes in Computer Science. Springer, 2019.

[91] Binhui Tang, Junfeng Wang, Huanran Qiu, Jian Yu, Zhongkun Yu, and Shijia Liu. Attack behavior extraction based on heterogeneous cyberthreat intelligence and graph convolutional networks. Computers, Materials & Continua, 2023.

[92] Wiem Tounsi. What is cyber threat intelligence and how is it evolving? Cyber-Vigilance and Digital Trust: Cyber Security in the Era of Cloud Computing and IoT, 2019.

[93] Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. arXiv preprint arXiv:2409.13373, 2024.

[94] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. CoRR, abs/2412.13663, 2024.

[95] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in NLP. In 14th International Conference on Computational Linguistics, COLING, 1992.

[96] Jason W. Wei and Kai Zou. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP. Association for Computational Linguistics, 2019.

[97] Ming Xu, Hongtai Wang, Jiahao Liu, Yun Lin, Chenyang Xu, Yingshi Liu, Hoon Wei Lim, and Jin Song Dong. Intelex: A llm-driven attack-level threat intelligence extraction framework. CoRR, abs/2412.10872, 2024.

[98] Jingchen Yan, Zhe Du, Jifang Li, Shiduo Yang, Jinghao Li, and Jianbin Li. A threat intelligence analysis method based on feature weighting and bert-bigru for industrial internet of things. Security and Communication Networks, 2022.

[99] Sarah Yoder. Automating mapping to att&ck: The threat report att&ck mapper (tram) tool. https://medium.com/mitre-attack/automating-mapping-to-attack-tram-1bb1b44bda76, 2019.

[100] Weiqiu You and Youngja Park. Cyber-attack technique classification using two-stage trained large language models. CoRR, abs/2411.18755, 2024.

[101] Yizhe You, Jun Jiang, Zhengwei Jiang, Peian Yang, Baoxu Liu, Huamin Feng, Xuren Wang, and Ning Li. TIM: threat context-enhanced TTP intelligence mining on unstructured threat data. Cybersecurity, 2022.

[102] Fengrui Yu and Yanhui Du. Few-shot learning of ttps classification using large language models. preprints.org, Jan 2024.

[103] Huixia Zhang, Guowei Shen, Chun Guo, Yunhe Cui, and Chaohui Jiang. Ex-action: Automatically extracting threat actions from cyber threat intelligence report based on multimodal learning. Security and Communication Networks, 2021.

[104] Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. RAFT: adapting language model to domain specific RAG. CoRR, abs/2403.10131, 2024.

[105] Yongheng Zhang, Tingwen Du, Yunshan Ma, Xiang Wang, Yi Xie, Guozheng Yang, Yuliang Lu, and Ee-Chien Chang. Attackg+:boosting attack knowledge graph construction with large language models. CoRR, abs/2405.04753, 2024.

[106] Jun Zhao, Qiben Yan, Jianxin Li, Minglai Shao, Zuti He, and Bo Li. Timiner: Automatically extracting and analyzing categorized cyber threat intelligence from social data. In Computers & Security, 2020.

[107] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Proceedings of Machine Learning Research. PMLR, 2021.

[108] Yue Zhou, Chenlu Guo, Xu Wang, Yi Chang, and Yuan Wu. A survey on data augmentation in large model era. CoRR, abs/2401.15422, 2024.

[109] Ziyun Zhu and Tudor Dumitras. Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In 2018 IEEE European Symposium on Security and Privacy, EuroS&P. IEEE, 2018.

Appendix

A Discussion on Implementations

As discussed in Sections 4.2, 5.2, and 6.2, our implementations capture the core, fundamental Natural Language Processing (NLP) methodologies adopted by state-of-the-art approaches. Note that studying individual tools falls out of the scope of this work since our goal is not to select a specific implementation instance, but to clarify the current limitations of the NLP methodologies. Our implementations do not capture additional components proposed by the literature, which are not part of the core NLP methodology, nor compositions of the aforementioned approaches.

NER. The core method is composed of the sequence of steps listed in Section 4.1, and for each of them, we consider the implementation specifically tailored for the Cyber Threat Intelligence domain. We exclude additional optimizations on the ontology class matching, as they are developed for custom ontologies (e.g., BM25 [38], or Entropy and Mutual Information [39], or custom algorithms [103]). Some works [58, 85, 86] include Relation Extraction (RE) – an NLP task that consists in identifying semantic relations among parts of text – to connect tokens related to TTPs into an "attack behavior graph". Studying knowledge graphs falls out of scope with respect to this work. Extractor [85] identifies attack actions through a custom gazetteer, and uses Semantic Role Labeling (SRL) to perform RE. AttacKG [58] includes a graph-alignment algorithm to match TTPs over a similarly constructed graph.

Classification. The identified core methodology involves classifying input sentences into ATT&CK Technique classes using fine-tuned BERT-based text classifiers [22], sentence transformer models [81], and data augmentation approaches (see Section 5.1). All selected works [6, 24, 37, 56, 77, 79, 80, 82, 98, 100] share this BERT-based methodology, differing mainly in minor changes or optimizations. To fairly compare their classification performance, we reimplement these core methods, intentionally omitting ad-hoc refinements such as additional layers (e.g., a Bidirectional Gated Recurrent Unit (BiGRU) layer, rarely included in the regular architecture in [98]), ontology-based embeddings (e.g., [37]), active learning strategies [77], post-processing algorithms to correct predicted labels [56], and multi-stage training processes [100]. Unlike TTPHunter [79], TTPXHunter [80], and ALERT [77], which use the *softmax* activation function in the classifier's

output layer, we use *sigmoid*, which is most commonly used in multi-label classification scenarios as the one presented in this work. Alam et al. [5] employ a similar unlabeled classifier architecture but compare embeddings at the sentence-slice level—rather than the full-sentence level—with a weighted sum of embeddings obtained separately from ATT&CK TTP titles and descriptions. Instead, we follow the established practice of generating embeddings by concatenating these two textual elements, aligning our methodology with other prominent studies [1,49]. Following a similar approach, we evaluate only the data augmentation optimization, as commonly proposed in the literature [35,45,80,100].

Generation. The main component consists of the LLM model, which is usually enriched via context in the prompt, through fine-tuning, or by making use of further processing steps.

All works use exactly our presented core methods or slight modifications. For example, Fieblinger et al. [26], AECR [14], Fengrui et al. [102] use the SFT in the same way as we do, except that AECR replaces the last layer of the tokenizer selection with a classification layer, which should lead to less hallucination. The methodology of Fengrui et al. corresponds to our "Raw" experiment in table 10. The same method of RAG is also used by Fayyazi et al. [24] and IntelEX [97], except that in IntelEX an LLM-as-a-judge is taken afterward to validate the results again for higher precision. FSP has the greatest variations, but also the most similar findings within the works. For example, CTINexus [15], Kumarasinghe et al. [49] and Fieblinger et al. [26] report instability problems or biases as reported in Section 6.2. While the last two works. [26,49] use FSP as we do, CTINexus [15] tries to solve the problem of static FSP entries by using a RAG system for FSP examples to avoid FSP bias. AttacKG+ [105] first tries to determine ATT&CK tactics and then enriches the prompt with all corresponding technique descriptions, which we classify as a "rudimentary" RAG system.

A.1 Comparison of Implementations

Table 13: Comparison of performances between related works and our corresponding implementations on AnnoCTR [50]. F1-Scores are calculated on increasing sets of TTPs from AnnoCTR.

		AnnoCTR		
Approach	**Method**	**Open-set Scenario (#TTPs)**		
		F1 (25)	**F1 (118)**	**F1 (637)**
NER	AttacKG [58]	42.78%	32.14%	23.75%
	Ours	59.91%	41.27%	35.55%
Classification	LADDER [5]	24.03%	22.34%	18.07%
	Ours	51.05%	33.96%	13.03%
Generation	aCTIon [88]	35.72 (± 3.67)%	22.52 (±2.54)%	8.65 (± 1.92)%
	Ours	43.00%	35.90%	25.90%

As mentioned in Section A, comparison of all individual state-of-the-art methods in a unified setting is, unfortunately, unfeasible due to frequent lack of data, code, and ontology. When those elements were available [14], we tested individual approaches and compared their performances with the underlying core methodology implementations. Since models and hyper-parameter values provided by original papers were tuned on different data, a fair comparison scenario can be provided using the "open-set scenario" experimental setup shown in Section 7. We use the original approaches to process input documents from the AnnoCTR dataset, obtain a list of techniques, and then evaluate the performances according to the metrics defined in Section 3.3. Interestingly, results in Table 13 show that optimizations proposed by original papers do not seem to generalize well to different data (and TTP) distributions: in most cases, our implementations have much higher performances. For example, the optimizations proposed by LADDER [5] seem to negatively impact the performances on both top-25 and top-118 common TTPs found in AnnoCTR documents. Most likely, such optimizations are too specialized for the specific testbed employed in the paper, often in the form of custom annotated datasets, and chosen sets of TTPs. Additionally, Table 13 shows that similarly to our results in Figure 4, a scaling behavior can be observed when increasing the number of TTPs, as well as higher average performances shown by traditional NER approaches. but it also evinces that our implementations are not representative of customized approaches. In conclusion, our results show that a direct comparison of these works with their customized optimizations is not appropriate, and emphasize the need to focus on the underlying core NLP methods in order to be able to make generalizable statements.

B Generation: Additional Experiments

We can extract the concrete MITRE concepts from the text output of generative LLMs by matching the concepts by their names or their IDs. Both ways give similar performances as shown in Table 14. As mentioned in Section 6.2, the LLM instruction datasets created from TRAM2 can be enriched

Table 14: Comparison of ID vs. Name matching at document and sentence level with generative LLMs with a raw prompt on AnnoCTR.

Level	**String-Matching**	**F1**	**Recall**	**Precision**
Document-Level	ID	0.267	0.262	0.314
	Name	0.270	0.271	0.329
Sentence-Level	ID	0.303	0.508	0.244
	Name	0.299	0.522	0.233

[14] Siracusano et al. [88] include in their paper all the original prompts and used models (GPT-3.5$_{\text{Turbo}}$) required to extract attack patterns, enabling us to reproduce their approach. The technique assignment hyper-parameter empirically found by the authors is not specified, so we calculate mean and standard deviation over 5 runs with increasing values.

Table 15: Comparison between TRAM2 RAG and the original TRAM2 datasets. Differences from the original TRAM2 dataset are in parentheses.

TRAM2			
Method	F1-Score (%)	Precision (%)	Recall (%)
SFT Raw	56.8 (-15.7)	54.6 (-25.4)	59.1 (-7.2)
SFT RAG	67.5 (+2.0)	68.2 (-7.8)	66.9 (+9.3)
SFT FSP+RAG	62.1 (-2.5)	69.4 (-7.0)	56.2 (+0.2)

Table 16: Test results of Llama3.3-70B-Instruct in comparison with Llama3.1-8B-Instruct (differences in parentheses).

TRAM2			
Method	F1-Score (%)	Precision (%)	Recall (%)
Raw	47.5 (-1.7)	38.9 (-0.7)	68.1 (+3.0)
RAG	51.3 (-2.8)	49.5 (+4.0)	59.6 (-7.2)

with RAG entries, based on an idea from Zhang et al. [104]. The final performances of fine-tuning with RAG and without are, however, quite similar as shown in Table 15. We also compared the significantly larger Llama3.3-70B-Instruct model with the performance of the Llama3.1-8B-Instruct model in Table 16. The same prompts were used without any adjustments. The results on the TRAM2 test data set are even slightly worse than with the smaller model. We can observe that the prompt (and the associated performance) is also strongly model dependent, even though it is a model from the same company and from the same LLama3 family.

C Hyper-parameter tuning of Classification Approaches

To investigate the impact of hyperparameter choices on model performance, we conducted a grid search analysis over five sets of hyperparameters using the $\text{RoBERTa}_{\text{Large}}$ model on the TRAM2 dataset. This involves re-evaluating the final test scores for each combination of hyperparameters, allowing us to identify which settings leads to the best results. Namely, the parameters are the batch size, the weight given to positive samples in the binary cross-entropy loss (Pos Weight), the learning rate (η), the decay of the learning rate (End Factor), and the number of *frozen* (i.e., non-trainable) encoding layers (Frozen Layers). In total, the same model is fine-tuned on 72 different combinations of such hyper-parameters. Figure 5 shows the best 15 combinations. Some hyper-parameters slightly decrease the performance. In particular, this happens when we let the learning rate decay to 0 during the training process, when assigning a weight to the positive class, and when using larger learning rates. Other hyper-parameters seem to have a small impact on the final results. In general, the impact of optimal hyper-parameters is small: the F1-score reaches an increment of 0.11%.

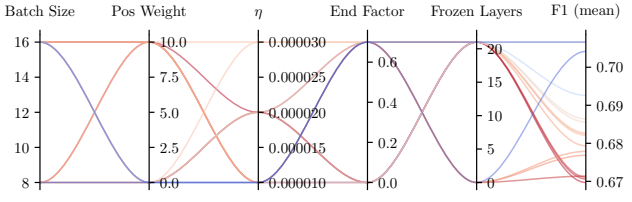

Figure 5: Parallel coordinates plot of the top-15 hyper-parameters combinations. In blue higher values of F1-Score, in red lower values.

D Models Tested

Table 17: Models used

Model	Parameters	Domain	Type
$\text{BERT}_{\text{Base, Uncased}}$	$1.10 * 10^8$	General	Classification (Encoder)
$\text{BERT}_{\text{Base, Cased}}$	$1.10 * 10^8$	General	Classification (Encoder)
$\text{RoBERTa}_{\text{Base}}$	$1.25 * 10^8$	General	Classification (Encoder)
$\text{XLM-RoBERTa}_{\text{Base}}$	$2.78 * 10^8$	General	Classification (Encoder)
$\text{RoBERTa}_{\text{Large}}$	$3.55 * 10^8$	General	Classification (Encoder)
$\text{XLM-RoBERTa}_{\text{Large}}$	$5.60 * 10^8$	General	Classification (Encoder)
DarkBERT	$1.25 * 10^8$	CTI	Classification (Encoder)
SecBERT	$8.35 * 10^7$	CTI	Classification (Encoder)
SecRoBERTa	$8.35 * 10^7$	CTI	Classification (Encoder)
CySecBERT	$1.10 * 10^8$	CTI	Classification (Encoder)
$\text{SciBERT}_{\text{Cased}}$	$1.10 * 10^8$	Scientific	Classification (Encoder)
$\text{SciBERT}_{\text{Uncased}}$	$1.10 * 10^8$	Scientific	Classification (Encoder)
CyBERT	$1.11 * 10^8$	CTI	Classification (Encoder)
SecureBERT	$1.25 * 10^8$	CTI	Classification (Encoder)
SBERT (All-MPNet-Base-V2)	$1.10 * 10^8$	General	Sentence Similarity
ATTACK-BERT	$1.10 * 10^8$	CTI	Sentence Similarity
$\text{SentSecBert}_{\text{10k}}$	$8.35 * 10^7$	CTI	Sentence Similarity
SFR-Embedding-2_R	$7.11 * 10^9$	General	Sentence Similarity
BAAI/bge-en-icl	$7.11 * 10^9$	General	Sentence Similarity
NV-Embed-V2	$7.85 * 10^9$	General	Sentence Similarity
Qwen2-7b	$7.07 * 10^9$	General	Generative (Decoder)
$\text{LLama-3.1}_{\text{8B}}$	$8.03 * 10^9$	General	Generative (Decoder)
$\text{LLama-3.3}_{\text{70B}}$	$7.00 * 10^{10}$	General	Generative (Decoder)

In Table 17, we provide a list of all the models we have tested.

E Closed-set Scenario with Increasing Labels

In this section, we compare NER, Classification, and Generation approaches on increasing amount of labels to identify. As in the Open-set Scenario (see Section 7), we select the AnnoCTR dataset as it has the highest number of (sub-)techniques. The tested models are respectively optimized on the label set. The final results in Figure 6 show that Classification approaches overperform other models when labeled data is available. Meanwhile, Generation approaches surpass NER models after 50 labels.

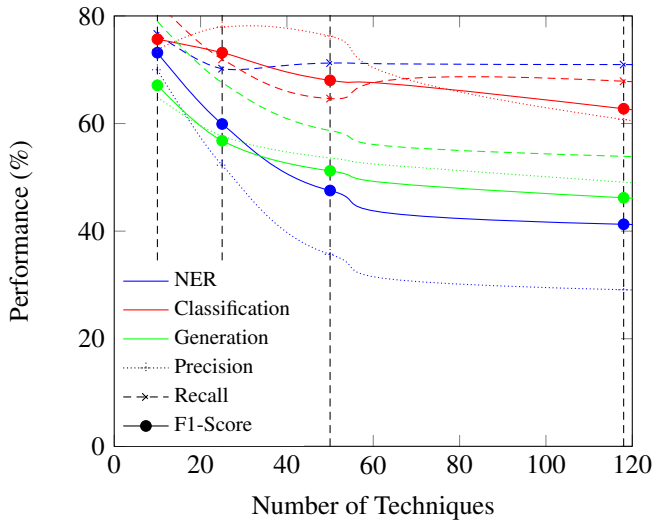

Figure 6: Comparison of performance on the top-k most common ATT&CK techniques for the AnnoCTR dataset with the best achieving approaches.

F Error Co-occurrences

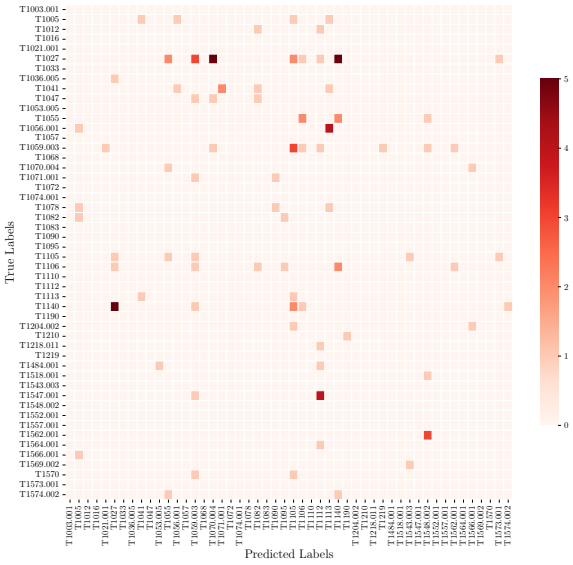

Figure 7: Error label co-occurrence matrix calculated on the TRAM2 test set.

In Figure 7, we show co-occurring labels on the TRAM2 test, calculated from $\text{RoBERTa}_{\text{Large}}$. On the y-axis, it shows the true labels appearing in the test samples; on the x-axis, it shows the erroneously predicted labels. Overall, it can be observed that many labels tend to co-occur more frequently than others, probably due to their similarity in meaning.