TTP Classification with Minimal Labeled Data: A Retrieval-Based Few-Shot Learning Approach

Dzenan Hamzic[1][0009−0008−4698−5534], Florian Skopik[1][0000−0002−1922−7892], Max Landauer[1][0000−0003−3813−3151], Markus Wurzenberger[1][0000−0003−3259−6972], and Andreas Rauber[2][0000−0002−9272−6225]

[1] AIT Austrian Institute of Technology, Vienna, Austria
firstname.lastname@ait.ac.at
[2] Vienna University of Technology, Vienna, Austria
rauber@ifs.tuwien.ac.at

Abstract. Cyber threat intelligence (CTI) reports are critical for understanding adversarial behaviors but are often unstructured and lack sufficient labeled data, which makes automated extraction of Tactics, Techniques, and Procedures (TTPs) challenging. This paper addresses these issues by introducing TTPFShot, a novel retrieval-based few-shot learning framework that classifies TTPs from CTI texts with minimal labeled data. TTPFShot leverages a vector database to retrieve semantically similar examples from a sentence-based dataset derived from the MITRE ATT&CK framework. These examples are used to construct few-shot prompts that guide large language models to accurately map CTI sentences to the appropriate TTP categories. Comprehensive evaluations on both sentence-based and document-based datasets demonstrate that TTPFShot outperforms existing approaches, such as TTPXHunter, by achieving higher precision, recall, and F1 scores. These results underscore the framework's ability to mitigate data scarcity issues and improve TTP classification accuracy in real-world settings.

Keywords: cyber threat intelligence · ttp classification · large language models · few-shot learning

1 Introduction

Cyber Threat Intelligence (CTI) reports provide critical information about how attackers perform cyberattacks, often including details like indicators of compromise (IoCs) — such as domain names, IP addresses, and Tactics, Techniques, and Procedures (TTPs) used by attackers. These reports help organizations better understand the methods and goals of cybercriminals [20,24]. Since they are primarily meant for human consumption, they are usually unstructured, written in everyday language, and put little effort in providing machine readable information.

An example of such a report is Trend Micro's "Pawn Storm's Lack of Sophistication as a Strategy" [15]. This report covers the tactics of the advanced

persistent threat (APT) group Pawn Storm, also known as APT28[3]. It describes how the group uses brute force techniques (T1110)[4] to break into accounts, either by guessing passwords or using password hashes they have obtained [3]. The report includes a detailed technical breakdown with code examples, IoCs, and related files, making it a valuable resource for spotting attack patterns. Many CTI reports from security companies, blogs, and forums use informal language, which can make it difficult to pull out useful information for automated security tools [18,25].

Organizations rely on these CTI reports to prepare for possible attacks. However, analyzing these reports to find useful information is a complex and time-consuming task for security analysts [25]. To understand attacks better and build stronger defenses, analysts often map the information from CTI reports to frameworks like MITRE ATT&CK. This framework organizes TTPs at different stages of an attack, such as initial access, execution, and persistence, and categorizes the goals of attackers, like stealing data or spying [19]. By using MITRE ATT&CK as a standard reference, organizations can understand the steps in an attack and evaluate their security measures [9,7].

Recognizing TTPs from CTI reports is challenging due to rapidly evolving attacker techniques, limited structured data, and language differences that complicate context and mapping to the MITRE ATT&CK framework [2,23]. Moreover, one-to-one mapping often misses relevant TTPs.

In this paper, we introduce **TTPFShot**, a framework designed to automatically extract TTPs from CTI texts. CTI reports are usually designed for human readers, thus they mostly lack structured data, making it hard to accurately classify the TTPs they describe in an automated manner at scale. TTPFShot tackles this issue by using Few-Shot Learning (FSL) [4], a method that requires only a small number of labeled examples, which is essential when labeled data is limited. This technique is efficient and enables our model to classify TTPs even with minimal labeled data by finding similar sentences that were extracted from labeled data and stored in a vector database.

The TTPFShot framework (Figure 1) works in several steps:

1. **COLLECT**: Extract sentences and corresponding labels from the MITRE ATT&CK framework.
2. **EMBED**: Embed the sentence and store the embeddings, along with their labels as metadata, in a vector database.
3. **QUERY**: Search the database for the N sentences that are most similar to the input sentence from a CTI report, and return their similarity scores.
4. **CLASSIFY**: Use the most similar sentences and their labels to create a few-shot prompt for the Large Language Model (LLM) to classify the new sentence according to the MITRE ATT&CK Technique or subtechnique.

This approach is especially innovative in TTP classification. Traditional models, like BERT (Bidirectional Encoder Representations from Transformers) [8],

[3] `https://attack.mitre.org/groups/G0007/`
[4] `https://attack.mitre.org/techniques/T1110/`

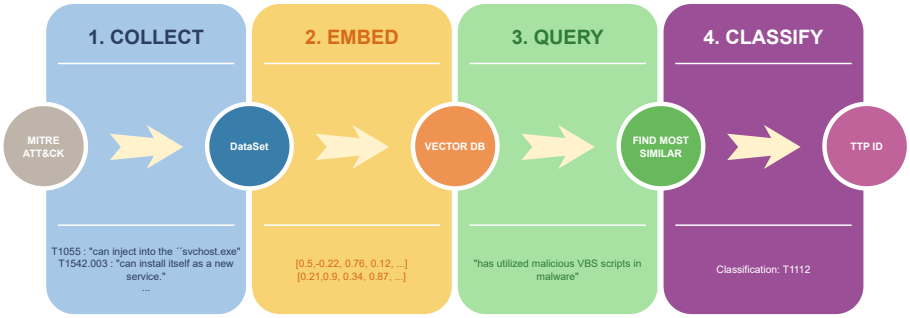


Fig. 1: TTPFShot Concept.

need a large number of labeled examples to perform well; for instance, TTPHunter notes that at least 50 sentences per TTP are required for fine-tuning a BERT model [22]. However, some TTPs in the MITRE ATT&CK framework have very few available examples. For example, the technique T1093 has only two sentences describing it, which would typically be insufficient for traditional models. TTPFShot, on the other hand, overcomes this limitation by relying on the vector database to supply additional context through similar examples, which the LLM can then use for FSL.

By making use of FSL in this way, TTPFShot significantly reduces the dependency on large datasets and offers a more flexible solution for classifying TTPs with minimal data. This framework has the potential to accelerate the process of analyzing CTI reports, making it easier for organizations to identify and respond to threats quickly and efficiently.

Our contributions collectively advance the state-of-the-art in TTP classification, addressing limitations in current approaches and offering resources and methods applicable to both research and practical cybersecurity applications. In detail, the main contributions of this paper are as follows:

1. A Novel Retrieval-Based FSL Approach for TTP Classification: We introduce an innovative approach that leverages retrieval-based FSL to classify TTPs, even when only a few labeled samples are available.
2. Extended Classification to MITRE ATT&CK Subtechniques: Unlike previous research, which primarily focuses on high-level TTPs, our approach extends to the classification of MITRE ATT&CK subtechniques.
3. Contribution of Two New Datasets: To facilitate TTP classification and further research, we publish two newly developed datasets: a sentence-based dataset (DSA)[5] and a document-based dataset (DSC)[6] from CTI reports.
4. Comprehensive Evaluation of TTPFShot: We perform an extensive evaluation of TTPFShot using the dataset provided by the authors of the leading approach, TTPXHunter [23]. Additionally, we benchmark TTPFShot

[5] https://zenodo.org/records/14907305
[6] https://zenodo.org/records/14659512

against TTPXHunter on our new document-based dataset DSC from CISA to compare performance in real-world settings.

5. Adaptation to One-to-Many Classification: Our approach introduces a one-to-many classifier, which improves upon the one-to-one TTP mapping in TTPHunter and TTPXHunter. For example, when a sentence such as "Payload scans system and network for system information and network connection discovery" is provided, TTPXHunter maps it to either T1082 (System Information Discovery) or T1049 (System Network Connection Discovery). However, TTPXHunter's one-to-one TTP mapping limits its capacity, as it only links each sentence to a single TTP, potentially missing other relevant TTPs in a document [22]. Our one-to-many classification approach overcomes this limitation by enabling the classification of multiple relevant TTPs from a single sentence, capturing a more comprehensive view of the ground truth TTPs within CTI reports.

The remainder of the paper is organized as follows. Sect. 2 provides background and related work on CTI and the challenges of TTP classification. It discusses existing tools such as TTPHunter, TTPXHunter, and TRAM, as well as the relevance of FSL and Retrieval-Augmented Generation (RAG) [13] in enhancing TTP classification. Sect. 3 outlines the datasets used in this study, including the DSC dataset from CISA, as well as the methodology of our TTPFShot approach, which combines vector databases, FSL, and LLMs for TTP classification. Sect. 4 describes the evaluation results of TTPFShot compared to other models, such as TTPXHunter, using both DSB and DSC datasets. Finally, Sect. 6 discusses the conclusions and future work.

2 Background and Related Work

In CTI reports[7][8], analysts find important information about new and ongoing cyber threats, including details on specific attacks and the TTPs attackers use. These reports often explain how attackers break into systems, stay hidden, steal data, and cover their tracks. They may also include Indicators of Compromise (IoCs), such as IP addresses or domain names, which signal possible threats [16].

For security analysts, CTI reports are essential tools for planning defense strategies. Analysts review TTPs in each report to see how these methods might affect their organization's setup and IT systems. For example, if a CTI report describes how attackers use techniques like credential dumping (e.g. T1003) or phishing (e.g. T1566) to gain access, analysts can check if their own detection systems are prepared to spot these threats [6].

By matching TTPs from CTI reports to their organization's systems, analysts can take measures to close security gaps and improve defenses. This may

[7] https://github.com/blackorbird/APT_REPORT/blob/master/BlindEagle/APT_Blind_Eagles_Malware_Arsenal_Technical_Analysis_of_the_New.pdf

[8] https://github.com/blackorbird/APT_REPORT/blob/master/lazarus/CryptoCore-Lazarus-Clearsky.pdf

involve updating detection rules, refining security policies, or training response teams on these attack methods. However, because CTI reports are often written in everyday language, it can be hard to quickly extract TTPs for automated tools. Tools that can automatically read and pull out TTPs from CTI reports are capable of reducing the workload for analysts and help organizations respond faster to new threats, but often fail to provide accurate results due to the following challenges [23]:

- Unstructured Language: CTI reports often use varied, everyday language, making it hard for automated tools to extract TTPs consistently [10].
- Rapidly Evolving Techniques: Attackers frequently modify or create new techniques, requiring models to stay up-to-date to recognize the latest TTPs [5].
- Limited Labeled Data: Many TTPs lack sufficient labeled examples, especially less common ones, making training difficult [22].
- Context-Dependent Classification: TTPs can mean different things based on context, requiring models to understand context to avoid misclassification [14,22].
- Multilabel Complexity: Actions often map to multiple TTPs, needing models that can handle multiple labels per instance [22].

These challenges underline the need for flexible approaches that can interpret unstructured text, adapt to new techniques, and classify TTPs with minimal data.

FSL is a machine learning approach designed for scenarios where only a small amount of labeled data is available. Unlike traditional models that require large datasets for effective training, FSL can classify new examples by learning from just a few instances per category. This makes it especially relevant for TTP classification, where labeled examples for each technique are often limited or sparse. In the context of CTI, some TTPs may only have a handful of examples, making traditional, data-hungry models like BERT less effective. However, a key limitation of retrieval-augmented FSL is that queries need to be sufficiently similar to the stored vectors in order to retrieve relevant examples. If the query differs too much from the existing data, the model might fail to recognize its similarity, leading to misclassifications. This can be problematic in other domains where the diversity of input data is large and unpredictable. In the case of CTI reports, however, this issue is somewhat mitigated. CTI texts often contain structured descriptions of common attack techniques, and the semantic similarity between sentences in these reports tends to be more predictable compared to other domains. As a result, FSL can still perform well in CTI applications despite this limitation, as the range of variations in the language used to describe TTPs is relatively narrow, and many of the techniques are described using similar phrasing or terminology across different reports.

TTPHunter [22], TTPXHunter [23], and TRAM [17] are tools developed to enhance the classification of TTPs in CTI domain. TTPHunter, for example, leverages BERT-based contextual feature extraction combined with a linear

classifier to map CTI sentences to specific TTP categories. While this method has proven effective in processing unstructured CTI reports, it faces limitations, such as its reliance on a substantial amount of labeled data and its inability to handle multi-label classification effectively. TTPXHunter builds on this by incorporating SecureBERT, which improves dataset augmentation through masked word prediction, helping to address the scarcity of labeled data and enhance the representation of underrepresented TTPs. However, its single-label classification approach still poses challenges when multiple TTPs are relevant in a single sentence. TRAM uses FastText embeddings and BERT-based models. It initially applies FastText with logistic regression for single-label classification of unstructured text, while multi-label classification is handled with multi-output logistic regression. For more complex tasks, TRAM uses BERT-based feature extraction, improving classification accuracy by capturing contextual nuances in sentences. However, BERT requires more labeled data and computational resources, limiting scalability. Despite these challenges, TRAM offers a flexible solution for TTP classification, dependent on data quality and feature extraction.

Our approach combines techniques from RAG and FSL to improve TTP classification in CTI. RAG operates in two parts: a retrieval step that searches a database to find relevant examples (sentences and corresponding TTP labels) and a generative model that uses these examples to make more accurate label predictions. In our method, the retrieval step is used to create or enhance the Few-Shot template. A Few-Shot template is a set of examples (sentences and their corresponding TTPs) that the model uses to understand the task with minimal labeled data. It retrieves sentences from the database that are similar to the input text and their associated TTPs, which serve as context for the classification task. These examples are then fed into a LLM, allowing it to classify the TTPs more accurately by leveraging the context provided in the Few-Shot template.

3 Methodology

In this section, we explain the methodology used to develop the design of our TTPFShot approach.

3.1 Retrieval Augmented Few Shot Template

In this study, we apply retrieval-augmented few-shot learning, a technique that combines retrieval-based methods with a prompting technique to classify CTI sentences into corresponding TTP ID labels. Figure 2 depicts the four steps of our approach and provides samples. In Step 1, the input sentence, such as "Payload scans system and network for system information and network connection discovery" is provided for classification. This sentence[9] describes an action that needs to be classified according to the MITRE ATT&CK framework. Step

[9] https://attack.mitre.org/techniques/T1049/

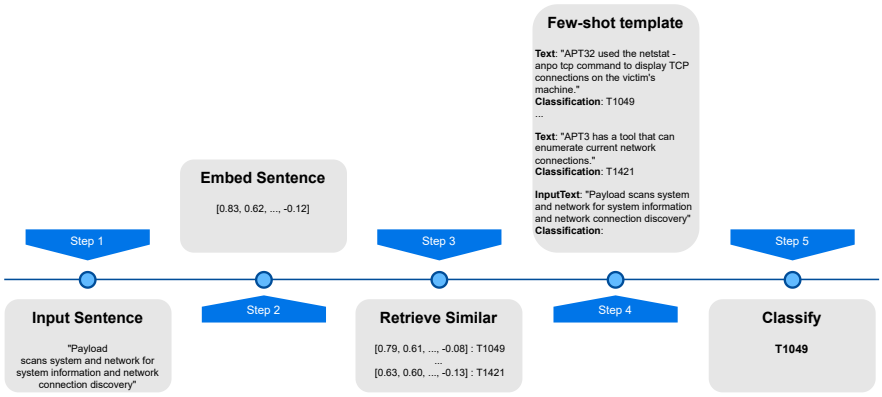


Fig. 2: Retrieval-based Few-Shot Templating.

2 involves converting this sentence into an embedding vector, which is a numerical representation that captures the semantic meaning of the sentence in a high-dimensional space. This transformation allows for better comparison between sentences. Step 3 follows, where a search is performed to retrieve similar sentence embeddings from a pre-existing dataset. Along with these similar vectors, their corresponding TTP IDs are returned. For example, similar sentences may be linked to TTP IDs such as T1049 and T1421, representing actions like "Network service scanning" and "Network connection enumeration" In Step 4, a few-shot template is constructed using the examples retrieved in Step 3, as well as the input sentence. These retrieved examples (such as "APT32 used the netstat -anpo tcp command to display TCP connections on the victim's machine" with classification T1049) are used to build a template that helps in classifying the input sentence. Finally, in step 5, the input sentence is classified based on the patterns learned from these few examples. The generative model uses these examples to make better predictions for the input sentence.

3.2 Data Sets

Sentence-based datasets, such as DSA and DSB, differ from document-based dataset DSC primarily in their granularity and structure. While document-based datasets contain full reports or articles with TTP references scattered throughout the text, sentence-based datasets break these reports down into individual sentences, each labeled with a specific TTP ID. This granular approach enables more precise classification, as each sentence is treated as a standalone instance for retrieval and prediction. In contrast, document-based datasets require models to understand broader context across the entire document, making them more complex and challenging to process due to the varying levels of relevance and context in different parts of the text. Sentence-based datasets, therefore, simplify

Table 1: Example TTP with description

TTP ID	Sentence
T1574.012	[Blue Mockingbird] has used `wmic.exe` and Windows Registry modifications to set the `COR_PROFILER` environment variable to execute a malicious DLL whenever a process loads the .NET CLR. (Citation: RedCanary Mockingbird May 2020)

Table 2: Rules, Examples, and Replacements

Rule	Example	Replacement
`re.sub(r'\[.*?\]\(https://attack.mitre.org/software/.*?\)', 'Software', text)`	[Some Software](https://attack.mitre.org/software/S1234)	Software
`re.sub(r'\[.*?\]\(https://attack.mitre.org/groups/.*?\)', 'Attacker', text)`	[Some Group](https://attack.mitre.org/groups/G1234)	Attacker
`df['Value'].str.replace(r'\([^)]*\)','',regex=True)`	This is a test (with some extra information)	This is a test
`df['Value'].str.replace(r'\[.*?\]', '', regex=True)`	This is a test [extra information]	This is a test
`df['Value'].str.strip()`	Extra spaces at start and end	Extra spaces at start and end

the task by focusing on smaller, more manageable chunks of information, which is especially beneficial for techniques like retrieval-augmented few-shot learning.

To build the DSA retrieval dataset, we utilized the MITRE ATT&CK framework, specifically version 15.1 of the enterprise data set[10]. From this framework, we extracted descriptions of techniques and examples of textual procedures associated with each technique. To improve the dataset's granularity, the textual descriptions were split into individual sentences, creating a sentence-based dataset. Each sentence was then automatically assigned to a specific technique or sub-technique ID, allowing for precise alignment and easier retrieval when querying the vector database (see example in Table 1).

Since most procedure description sentences contain group names (replaced with Rule 2), software names (replaced with Rule 1), corresponding URLs (replaced with Rules 1 and 2), and brackets (replaced with Rules 3 and 4), the preprocessing rules in Table 2 were applied to clean and standardize the data. Rule 5 removes the extra spaces at start and end. The DSA dataset includes a total of *780 unique techniques and sub-techniques*, significantly expanding the number of labels compared to the 193 technique labels in TTPXHunter and the 50 techniques in TRAM. Overall, the dataset contains 19,747 rows. As can be seen from the Figure 3, the highest number of sentences are associated with the following TTPs: T1105, T1082, T1071.001, T1059.003, and T1083, with sentence counts of 468, 387, 349, 345, and 324, respectively. On the other hand, the TTPs

[10] `https://raw.githubusercontent.com/mitre-attack/attack-stix-data/refs/heads/master/enterprise-attack/enterprise-attack-15.1.json`

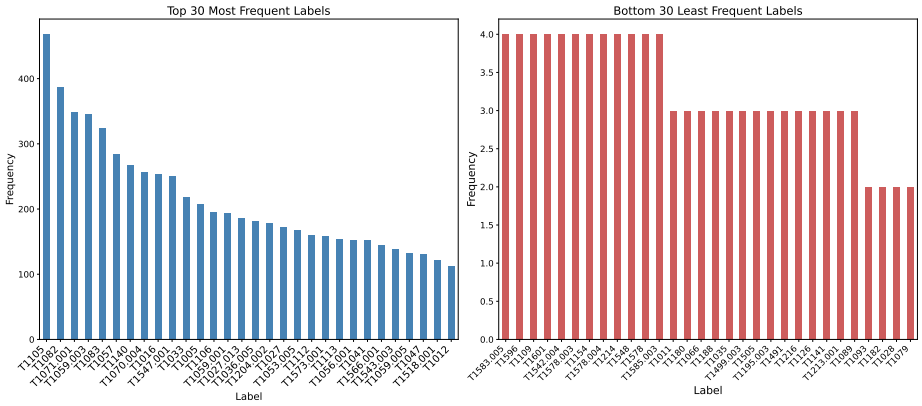


Fig. 3: Label Distribution.

with the fewest sentences are T1093, T1079, T1028, T1182, and T1195.003, each with only 2 or 3 sentences.

We used the sentence-based TTPHunter dataset[11] (DSB) to determine the optimal number of examples (shots) needed for FSL and to perform an initial comparison with the current leading text-to-TTP classification framework, TTPXHunter. The dataset uses the standard MITRE ATT&CK knowledge base as a data source to collect text data and their corresponding TTPs. The Indicators of Compromise (IOC) are replaced with their base names, e.g., the email adress `xyz@pqr.com` is replaced with the word "Email". This dataset consists of 8,387 rows, with two main columns: TTP ID and Sentence. It includes only 50 unique TTP IDs, selected by the TTPHunter authors based on the availability of at least 50 sentences for each TTP, covering 177 unique MITRE ATT&CK techniques [22]. The dataset preparation methodology differs from ours (DSA dataset), since we did not remove the IOCs, and we included TTP labels having less then 50 sentences, i.e., we included all TTP labels.

The DSC dataset comprises 77 cybersecurity articles sourced from the public CISA website, published between July 2020 and February 2024. Each article explicitly mentions MITRE ATT&CK Tactics, Techniques, and Procedures (TTPs), providing valuable insights into adversary behaviors. The dataset includes the following columns:

- RawText: The unfiltered text extracted from the main content of each article.
- TTP: A set of MITRE ATT&CK TTP IDs found within the article's RawText. This column serves as a ground truth for multilabel classification.
- CleanText: A cleaned version of the RawText, with tables and TTP IDs removed for clarity. Sentences from this column serve as input to be classified.

[11] https://github.com/nanda-rani/TTPHunter-Automated-Extraction-of-Actionable-Intelligence-as-TTPs-from-Narrative-Threat-Reports/blob/main/Dataset/TTPHunter_dataset.csv

– URL: The URL to the original article.

To standardize the text and facilitate analysis, TTP mentions were extracted and replaced with empty spaces using the regex pattern:

```
(?:TA\d{4}|T\d{4,5}(?:\.\d{3})?)
```

The extracted TTP IDs serve as labels or ground-truth of the corresponding document. This step is crucial as it removes the ground truth labels from the text, effectively preventing the model from directly relying on these labels during classification. By doing so, we ensure that the classification algorithms are evaluated purely on their ability to infer TTPs from the context of the report, rather than using predefined labels that may influence the outcome. This approach helps in assessing the model's actual performance in detecting and classifying TTPs without bias from the known ground truth labels. The sentences which contain no TTP labels are left intact.

3.3 Classification Approach Design

The workflow for our TTP classification system (Figure 4) starts with preparing a dataset of sentences from a DSA, each labeled with its corresponding TTP ID (TTP Sentences). These sentences, which describe specific TTPs used in cyber-attacks, are then processed into embeddings using SecureBERT [1]. SecureBERT generates vector embeddings with a dimension of 1x768 for each sentence, capturing its semantic meaning. These embeddings are stored in a vector database (or "vector DB").

When a user or system sends a new sentence (e.g., from a CTI report that needs TTP labeling) for classification, the sentence is first embedded using the same SecureBERT model to ensure consistency with the stored embeddings. This new sentence embedding is then used to query the vector store, retrieving the N most similar sentences based on semantic similarity. These retrieved sentences provide relevant context and examples. The FSL template is created by organizing these examples in a structured format that helps the model make accurate predictions. This template plays a crucial role in guiding the model by providing relevant context to classify the new sentence effectively.

The Few-Shot Template, now containing the N similar sentences as examples, is sent as a prompt to a LLM. The LLM processes this prompt and classifies the input sentence by predicting the appropriate TTP ID. The output returned by the LLM is solely the TTP ID label (or multiple TTP IDs, depending on the prompt setting), which represents the best match for the queried sentence based on the examples provided in the Few-Shot template.

Querying on a sentence level is more effective relative to whole document querying because it allows for precise and granular classification, focusing on specific TTP descriptions that are clearly defined within individual sentences. This approach reduces ambiguity and makes it easier to match the sentence to the correct TTP ID, as each sentence is a self-contained unit with a clear context.

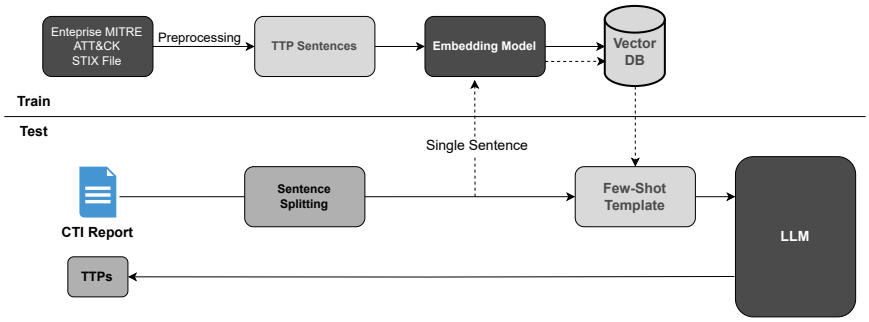


Fig. 4: TTPFShot Concept Overview.

Since the dataset for retrieval is sentence-based, the model is optimized for this level of granularity, whereas whole document querying could introduce noise and require more complex context handling.

3.4 Configuration of TTPFShot's Few Shot Template

In our experiment, we simulated varying numbers of Few-Shot samples (FS number) to evaluate their impact on the model's performance metrics: F1 Score, Precision (Micro), and Recall (Micro). Few-Shot samples refer to a small number of labeled examples used by the model to learn and make predictions. Figure 5 illustrates these metrics across different Few-Shot numbers, showing how the model's performance stabilizes as we increase the number of Few-Shot samples. For this evaluation, we used a dataset split with 80% for training and 20% for testing. The 80/20 dataset split is a commonly used approach in machine learning, as it provides a good balance between training the model on a sufficient amount of data (80%) while retaining enough data (20%) for testing, ensuring effective evaluation of the model's generalization capability [22].

We chose the number 65 as the parameter setting for the number of shots in Few-Shot algorithm because it provides a balance between performance and computational efficiency. At 65 (Figure 5), the micro F1 score, precision, and recall show optimal or near-optimal values, indicating a high level of accuracy and consistency. This ensures that the model performs well without excessive computational costs or diminishing returns at higher few-shot numbers.

3.5 Few-Shot Template

The Few-Shot template (Figure 6) is designed to help the LLM classify CTI sentences into specific MITRE ATT&CK TTP categories. This template includes 65 examples of TTP-labeled sentences, retrieved using a vector retriever, providing context to guide the model's classification decision. These 65 examples are the most similar to the input text that is to be classified. The similarity of

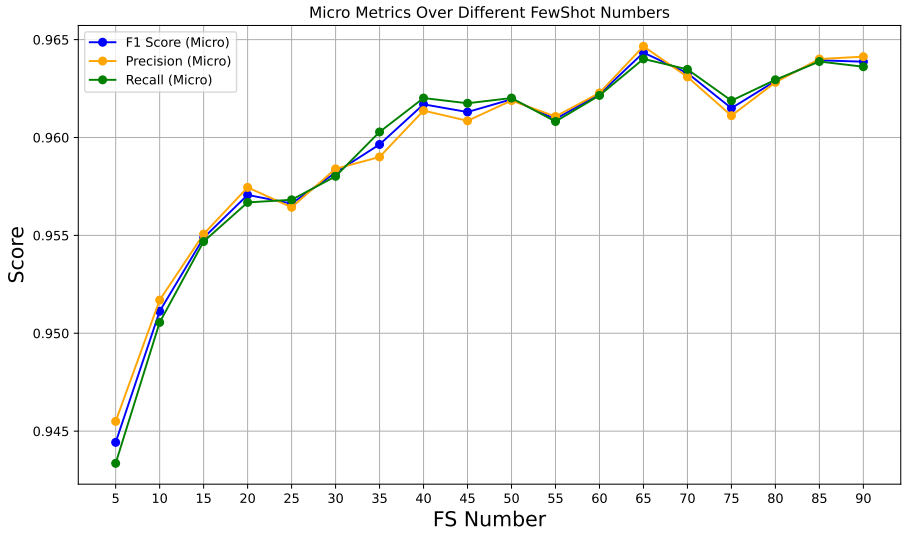


Fig. 5: Evaluation of Varying Few-Shot numbers on Prediction Performance.

each example to the input text (query) is given by the retriever, measured by the L2 distance in float, with a lower score representing higher similarity. Each example consists of a sentence, a similarity score, and its corresponding TTP classification label. The template begins by instructing the LLM that it is acting as a "MITRE ATT&CK TTP classification expert." Its task is to classify an input text based on examples provided in the template. Each example follows a structured format:

- **Text**: The sentence itself, which describes a specific tactic, technique, or procedure observed in a cyberattack.
- **Similarity**: The similarity score between the example sentence("Text") and the input sentence ("InputText"). This score indicates how closely the example matches the input, with lower values representing higher similarity.
- **Classification**: The TTP label that corresponds to the example sentence. This label starts with "T" and is followed by a unique identifier number from the MITRE ATT&CK framework (e.g., T1082).

If the LLM finds that the "InputText" is not related to any of the examples provided, it is instructed to return the label "T0000". The similarity score acts as a threshold, guiding the model to avoid assigning a TTP label if the input text does not sufficiently match any example.

Below is the properly formatted template (Figure 6), showing a sample of the structure. Note that in the actual implementation, 65 examples would be included, providing a robust reference set for the LLM to classify the input sentence accurately.

```
You are a MITRE ATT&CK TTP classification expert. Your task is to classify the
'InputText' based on the provided examples below. Each example shows:
- a sentence (InputText),
- inverse similarity to InputText (the lower the better) (Similarity),
- and its corresponding classification label starting with 'T'.
Use these examples to determine the correct classification for the given text
(InputText). If 'InputText' is completely not related to any of the provided examples,
return class 'T0000'. To determine if the 'InputText' is not related to the given
examples, you can use the 'Similarity' property of the examples. Low 'Similarity'
values indicate high similarity between strings.Return only the classification label
starting with 'T' or 'T0000' if you found no appropriate class for 'InputText'.

Text: tajmahal has the ability to identify hardware information, the computer name,
and OS information on an infected host.[1]
Similarity: 0.09965219
Classification: T1082

Text: runningrat gathers the OS version, logical drives information, processor
information, and volume information.[1]
Similarity: 0.11707238
Classification: T1082
...
InputText: reaver collects the victim's IP address.[1]
Classification:
```

Fig. 6: TTPFShot Prompt Template.

4 Evaluation

This section presents a comprehensive evaluation of TTPFShot and competing methods across both sentence-based (DSB) and document-based (DSC) datasets, utilizing a diverse set of metrics—including accuracy, precision, recall, F1-score, Hamming accuracy, as well as novel measures such as coverage and bad coverage—to capture the nuances of multilabel classification in cybersecurity threat intelligence.

4.1 Evaluation Metrics

As previously mentioned, TTPFShot is evaluated on two distinct datasets: a sentence-based dataset (DSB) and a document-based dataset (DSC). For the DSB, we consider the test set as the ground truth for comparison. The DSC dataset, however, consists of a comprehensive crawl from the CISA website, featuring Cybersecurity Alerts & Advisories articles that have been manually annotated with TTP IDs. This dataset serves as a rigorous benchmark to evaluate the performance of our approach alongside TTPXHunter.

To assess the models' performance comprehensively, we employ several evaluation metrics: Accuracy [26], Precision [11], Recall [21], F1-score [27], as well as Hamming Accuracy [12]. The latter is particularly useful for evaluating multilabel classification in the DSC dataset. These metrics help us gauge the effectiveness of TTPFShot in correctly identifying relevant TTPs within each document. To provide additional insights into the model's ability to perform multilabel clas-

sification tasks, we propose two novel metrics called Coverage and Bad Coverage, which we define and explain in the following.

Coverage measures how much of the ground truth is correctly captured by the predicted labels (see Eq. 1). It calculates the proportion of elements in the ground truth that are also present in the predicted set. In other words, it tells us how well the model covers the correct answers.

$$\text{Coverage Ratio} = \frac{|y_{\text{true}} \cap y_{\text{pred}}|}{|y_{\text{true}}|} \tag{1}$$

Bad Coverage measures how much of the prediction is incorrect or missing when compared to the ground truth (see Eq. 2). It calculates the proportion of elements in the predicted set that do not match or are missing from the ground truth. Essentially, this metric highlights the errors or omissions in the model's predictions.

$$\text{Bad Coverage Ratio} = \frac{|y_{\text{pred}} \setminus y_{\text{true}}|}{|y_{\text{pred}}|} \tag{2}$$

4.2 TTPFShot Evaluation

In this section, we present an evaluation of TTPFShot's performance on both the sentence-based DSB (single-label) dataset and the DSC (multilabel) dataset.

4.3 Evaluation on DSB Dataset (Sentence-based)

Figure 7 illustrates the evaluation results of TTPFShot, TTPHunter, and TTPX-Hunter on DSB across three metrics: Precision, Recall, and F1-Score. The scores demonstrate that TTPFShot consistently outperforms the other two methods, achieving the highest values (0.96 for all three metrics) compared to TTPHunter (Precision: 0.89, Recall: 0.88, F1-Score: 0.88) and TTPXHunter (Precision: 0.94, Recall: 0.92, F1-Score: 0.92). This highlights the effectiveness of default setting (SL - Single Label) TTPFShot, which uses the GPT-4o model, in accurately extracting and classifying TTPs from CTI sentences on the DSB dataset.

4.4 Evaluation on DSC Dataset (Document-based)

Evaluation on DSC (Table 3) is more challenging and reflects real-world CTI reports. The abbreviations next to TTPFShot represent different configurations: "Multi-Label" (ML), "Zero-Shot" (ZS), "Single Label LLaMA 3 70B-Instruct" (SL L), and "Single Label Mixtral 7x8B-Instruct v0.1" (SL M).

The following configurations of TTPFShot were evaluated, each with different settings for handling the classification task:

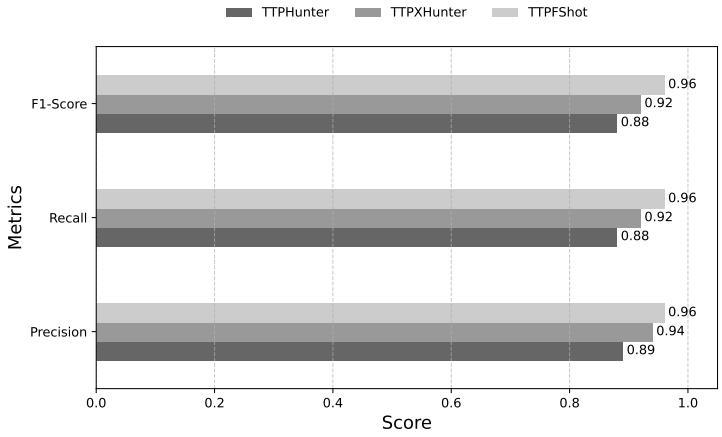


Fig. 7: Performance Comparison on DSB Dataset.

Table 3: Evaluation Metrics for Models on DSC Dataset

Model	Hamming Acc.	F1	Precision	Recall	Jaccard	Coverage	B. Coverage
TTPXHunter	0.91	0.09	0.13	0.10	0.06	0.11	0.85
TTPFShot SL	0.92	0.28	0.19	0.35	0.14	0.38	0.78
TTPFShot ML	0.93	0.26	0.16	0.35	0.12	0.37	0.81
TTPFShot ZS	0.91	0.17	0.16	0.21	0.10	0.24	0.80
TTPFShot SL L	0.82	0.08	0.09	0.25	0.07	0.26	0.87
TTPFShot SL M	0.92	0.24	0.15	0.30	0.11	0.33	0.82

– **Multi-Label (ML):** In this setting, TTPFShot's prompt was modified to include a sentence instructing the model to output multiple labels starting with "T" if applicable. This setting uses OpenAI's GPT-4o model.
– **Zero-Shot (ZS):** In this configuration, TTPFShot's prompt was adjusted to exclude any Few-Shot examples. This setting uses GPT-4o model as well.
– **Single Label LLaMA 3 70B-Instruct (SL L):** TTPFShot was executed using the LLaMA 3 70B-Instruct model instead of the ChatGPT 4o model.
– **Single Label Mixtral 7x8B-Instruct v0.1 (SL M):** TTPFShot was executed using the Mixtral 7x8B-Instruct v0.1 model instead of the ChatGPT 4o model.

Table 3 shows that while TTPXHunter achieves a high Hamming Accuracy (0.91), its F1 (0.09), Precision (0.13), Recall (0.10), and Jaccard (0.06) scores are very low, with a Coverage of only 0.11 and a high Bad Coverage of 0.85—indicating many missed or misclassified TTPs. In contrast, TTPFShot SL improves recall (0.35) and F1 (0.28), with better Coverage (0.38) and lower Bad Coverage (0.78), suggesting it more effectively captures true positives while reducing errors. Similarly, TTPFShot ML achieves the highest Hamming Accuracy (0.93) with comparable recall (0.35) and balanced Coverage (0.37) and Bad Coverage (0.81). The Zero-Shot variant (TTPFShot ZS) and the single-label models based on LLaMA 3 (SL L) and Mixtral (SL M) show lower overall perfor-

Table 4: Evaluation Metrics for Models using Subset of Data

Model	Hamming Acc.	F1	Precision	Recall	Jaccard	Coverage	B. Coverage
TTPXHunter	0.78	0.36	0.36	0.32	0.20	0.29	0.58
TTPFShot SL	0.75	0.50	0.38	0.62	0.31	0.65	0.56
TTPFShot ML	0.74	0.48	0.36	0.62	0.30	0.65	0.57
TTPFShot ZS	0.76	0.46	0.38	0.55	0.29	0.58	0.55
TTPFShot SL L	0.60	0.24	0.22	0.50	0.18	0.49	0.70
TTPFShot SL M	0.72	0.45	0.33	0.58	0.27	0.58	0.61

mance—especially SL L with an F1 of 0.08, Coverage of 0.26, and the worst Bad Coverage (0.87). Overall, TTPFShot SL and ML offer the best trade-offs, delivering higher recall and coverage with fewer incorrect predictions compared to the alternatives. The F1 scores across all models are relatively low (Table 3), reflecting the inherent challenges in achieving a balance between precision and recall in this complex task. While F1 provides a valuable summary of the trade-off between these two metrics, it alone does not fully capture the nuances of model performance, particularly in real-world CTI scenarios (DSC dataset). Similar to TTPHunter, where document-based evaluation (F1-score: 0.75) performed worse than sentence-based evaluation (F1-score: 0.88), we observe a performance drop when evaluating on DSC compared to DSB. This may be due to the polymorphic nature of words—whose meanings vary with context in CTI reports often marked by broad, vague descriptions—and language differences that further complicate TTP identification [22,2,23].

Refining the Evaluation to only Important Techniques without Subtechniques We refined the scope of the evaluation to focus exclusively on techniques, excluding subtechniques, to streamline the task and improve consistency. Furthermore, we prioritized important or more frequently referenced techniques by limiting the evaluation to those with over 50 sentences in their MITRE ATT&CK descriptions. This ensures that the analysis concentrates on widely recognized techniques with sufficient descriptive context, enhancing the relevance and robustness of the evaluation.

Table 4 shows that although TTPXHunter achieves the highest Hamming Accuracy (0.78), its overall performance is weaker—particularly in the Coverage metric (0.29), which indicates that it captures only a small portion of the ground truth labels in the documents. In contrast, TTPFShot SL stands out with the highest F1 score (0.50) and recall (0.62), paired with the best coverage (0.65) and low bad coverage (0.56)—indicating it not only captures more of the relevant TTPs but also minimizes incorrect predictions. TTPFShot ML performs similarly, effectively handling multi-label cases, whereas the zero-shot version and alternative single-label variants (SL L and SL M) lag behind, particularly in coverage quality and error rates.

Overall, the results indicate that TTPFShot SL and TTPFShot ML offer the best trade-offs between various metrics. Their performance makes them particularly well-suited for real-world CTI applications that require an efficient balance between precision, recall, and multi-label classification.

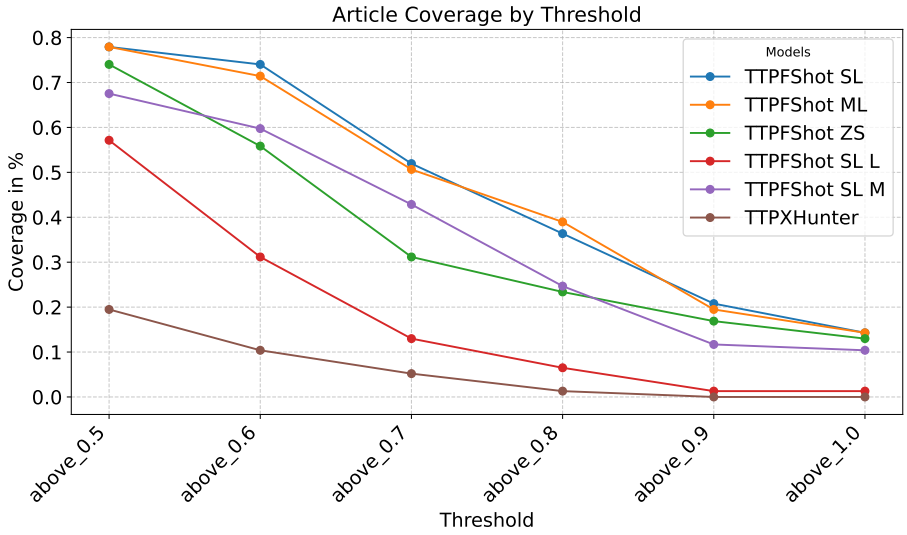
To gain a more nuanced understanding of the model's performance, additional metrics such as coverage and bad coverage were introduced, providing valuable insights into how well the models capture relevant TTPs and avoid incorrect predictions in multilabel classification tasks. This approach ensures a balanced evaluation that goes beyond traditional metrics.

Figure 8a shows how coverage varies across models as the threshold for matching ground truth increases (from 0.5 to 1.0). As the threshold increases, fewer labels are assigned because the retrieved examples are not similar enough to the input sentence. TTPFShot SL consistently achieves the highest coverage, maintaining strong performance even at stricter thresholds. TTPFShot ML and TTPFShot ZS perform competitively but fall short of TTPFShot SL, especially at higher thresholds. TTPXHunter starts strong at lower thresholds but declines significantly as the threshold tightens. Single-label variants, TTPFShot SL L and SL M, exhibit the lowest coverage, struggling to match the performance of other models. Overall, TTPFShot SL proves to be the most robust and reliable model across varying threshold levels. Figure 8b highlights the proportion of incorrect or irrelevant predictions across models as the threshold for matching ground truth increases (above 0.5 to 1.0). TTPFShot SL exhibits the lowest bad coverage across all thresholds, indicating minimal errors in its predictions even under stricter conditions. TTPXHunter and TTPFShot ZS have higher bad coverage, particularly at lower thresholds, but improve slightly as thresholds tighten. In contrast, TTPFShot SL L and SL M show the highest bad coverage, reflecting significant prediction errors compared to other models.

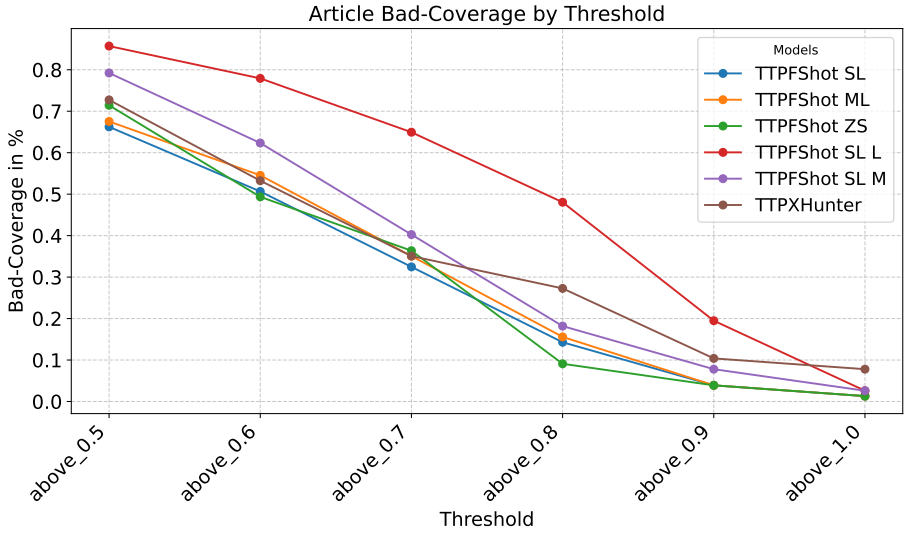
Unlike traditional metrics like F1-score, Accuracy, Precision, and Recall, which assess classification correctness at the sample level, Coverage and Bad Coverage offer a broader view of how well a model retrieves relevant TTPs while minimizing incorrect predictions. The coverage and bad coverage plots (see Fig.8a and Fig.8b) highlight these aspects, capturing the percentage of ground truth retrieved and the proportion of incorrect classifications—insights not provided by Recall, which ignores false positives, or F1-score, which balances Precision and Recall but does not distinguish between missing relevant TTPs (low Coverage) and assigning incorrect labels (high Bad Coverage). TTPFShot SL emerges as the most reliable model, achieving high coverage with minimal bad coverage across all thresholds.

5 Discussion

Our experiments with TTPFShot highlight both the strengths and limitations of using a retrieval-augmented few-shot learning approach for TTP classification. In early implementations, TTPFShot employed standard BERT embeddings; however, evaluations on the DSB dataset revealed that SecureBERT embeddings significantly enhance precision, recall, and F1 scores by better capturing cybersecurity-specific language. Efforts to improve retrieval via vector database partitioning did not yield further benefits (see Sect. 3.1). We are aware that LLMs struggle with floats, but we found out that including L2 distance in few-



(a) Coverage



(b) Bad Coverage

Fig. 8: Comparison of Coverage and Bad Coverage.

Table 5: Mean and Standard Deviation of document processing times (in minutes) for each tool

Model	Mean	Std
TTPFShot SL	3.2	1.5
TTPFShot ML	2.9	1.3
TTPFShot ZS	1.8	0.8
TTPXHunter	0.7	0.4

Table 6: Comparison of Pros and Cons: TTPFShot vs. TTPXHunter

Aspect	TTPFShot	TTPXHunter
Pros	Minimal labeled data requirement; supports one-to-many (multi-label) classification; no fine-tuning needed; easily updatable retrieval dataset.	Fast execution speed.
Cons	Slower processing time; higher computational cost; dependent on internet latency and shared LLM resources.	Requires time-intensive fine-tuning upon framework updates; requires time-intensive dataset construction;.

shot template as "Similarity" improves classification in our case. For the few-shot classification stage, we evaluated several OpenAI models. Notably, models such as `gpt-4o-mini` and `o1-preview` underperformed relative to `gpt-4o`, which demonstrated superior generalization from few-shot examples. Consequently, `gpt-4o` was adopted as the default model in TTPFShot.

Execution performance is a key consideration for real-world CTI applications. We measured the average sentence classification times on the DSC dataset for different TTPFShot configurations, as compared to TTPXHunter. Table 5 summarizes these results. The results indicate that TTPFShot in both single-label and multi-label modes is notably slower than TTPXHunter. Although the zero-shot variant improves speed, it does so at the expense of classification performance (see Table 3 and Table 4). The slower processing time of TTPFShot is largely due to reliance on online LLM APIs and internet latency; with dedicated LLM instances, we expect execution times to improve.

A side-by-side comparison of TTPFShot and TTPXHunter is provided in Table 6. TTPFShot's main advantages include its minimal need for labeled data, support for one-to-many (multi-label) classification, elimination of fine-tuning, and ease of updating the retrieval dataset when the MITRE ATT&CK framework is revised. In contrast, TTPFShot's drawbacks are its slower processing speed and higher computational cost. TTPXHunter benefits from fast execution and simplicity but requires time-intensive fine-tuning to adapt to framework updates.

In summary, TTPFShot is well-suited for scenarios with scarce labeled data and where multi-label classification is essential, offering flexibility and ease of

updates. However, its slower processing speed and higher operational costs may limit its applicability in real-time settings, where TTPXHunter's rapid execution can be advantageous despite its fine-tuning and dataset construction overhead. As for the practical dimension, TTPFShot will be integrated into Taranis AI[12], where it will operate as an asynchronous bot that periodically processes newly collected CTI data, showcasing its potential for real-world deployment.

6 Conclusion

In this paper, we introduced TTPFShot, a novel retrieval-based few-shot learning framework for classifying TTPs in CTI reports. TTPFShot leverages a vector database to retrieve semantically similar examples from a labeled sentence-based dataset (DSA), which are then used to construct a Few-Shot template for an LLM-based classification model. This approach allows TTP classification with minimal labeled data, making it highly effective for cybersecurity applications where annotated datasets are scarce.

To validate its real-world applicability, we evaluated TTPFShot using DSC, a real-life CTI dataset containing cybersecurity reports from CISA. The framework demonstrated its capability to classify 780 MITRE ATT&CK techniques and subtechniques, significantly expanding the scope of previous approaches. We conducted a comparative evaluation against existing models like TTPHunter and TTPXHunter, using multiple datasets (DSB and DSC) and standard classification metrics such as F1-score, Precision, Recall, and Hamming Accuracy. In addition, we evaluated the approaches with two novel metrics proposed in this paper, namely Coverage and Bad Coverage, which measure the proportions of correct answers and errors respectively.

The evaluation showed that TTPFShot outperforms TTPXHunter and other baselines, particularly in multi-label classification and recall, indicating higher effectiveness in capturing relevant TTPs from unstructured CTI texts. Our results highlight the advantages of retrieval-augmented few-shot learning over traditional supervised approaches, reducing dependence on large labeled datasets while improving classification accuracy.

Future work will explore enhancing retrieval mechanisms, optimizing classification prompts, and integrating more cybersecurity-specific embeddings to further improve performance and adaptability.

Acknowledgments

Funded by the European Union under the European Defence Fund (GA no. 101121403 - NEWSROOM and GA no. 101121418 - EUCINF). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

[12] `https://taranis.ai/`

This work is co-funded by the Austrian FFG Kiras project ASOC (GA no. FO999905301).

References

1. Aghaei, E., Niu, X., Shadid, W., Al-Shaer, E.: Securebert: A domain-specific language model for cybersecurity. In: Security and Privacy in Communication Networks: 18th EAI International Conference, SecureComm 2022. pp. 39–56 (2023)
2. Alam, M.T., Bhusal, D., Park, Y., Rastogi, N.: Looking beyond iocs: Automatically extracting attack patterns from external cti. arXiv (2022), https://arxiv.org/abs/2211.01753
3. ATT&CK, M.: Brute force, technique t1110 - enterprise. https://attack.mitre.org/techniques/T1110/ (nd)
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Amodei, D.: Language models are few-shot learners. Advances in Neural Information Processing Systems **33**, 1877–1901 (2020)
5. Chen, F., Wu, T., Nguyen, V., Wang, S., Hu, H., Abuadbba, A., Rudolph, C.: Adapting to cyber threats: A phishing evolution network (pen) framework for phishing generation and analyzing evolution patterns using large language models. arXiv (2024), https://arxiv.org/abs/2411.11389
6. CISA: Best practices for mitre att&ck mapping. https://www.cisa.gov/sites/default/files/2023-01/Best%20Practices%20for%20MITRE%20ATTCK%20Mapping.pdf (2023)
7. to Knowledge: Tackling Challenges in Cyber Threat Intelligence, F.N.: Picus security. https://www.picussecurity.com/resource/blog/from-noise-to-knowledge-tackling-challenges-in-cyber-threat-intelligence (2023)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies **1**, 4171–4186 (2019). https://doi.org/10.18653/v1/N19-1423
9. DISA: The mitre att&ck framework: Using ttps to understand cyber threats. https://www.disa.mil/mitre-attack (2023)
10. Fieblinger, R., Alam, M.T., Rastogi, N.: Actionable cyber threat intelligence using knowledge graphs and large language models. arXiv (2024), https://arxiv.org/abs/2407.02528
11. Flach, P.A.: Precision-recall and roc curves. Machine Learning **68**(2), 227–243 (2006). https://doi.org/10.1007/s10994-006-6676-4
12. Hamza, W., Zhang, Y.: Hamming accuracy for classification models in multi-class problems. Journal of Computing Research **4**(3), 24–32 (2013). https://doi.org/10.1016/j.jcr.2013.04.002
13. Lewis, P., Piktus, A., Karpukhin, V., Oguz, B., Min, S., Wu, L., Riedel, S.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
14. Mavroeidis, V., Hohimer, R., Casey, T., Jøsang, A.: Threat actor type inference and characterization within cyber threat intelligence. 13th International Conference on Cyber Conflict (CyCon) (2021), https://doi.org/10.48550/arXiv.2103.02301

15. Micro, T.: Pawn storm uses brute force and stealth against high-value targets (2024)
16. Microsoft: What is cyber threat intelligence? https://www.microsoft.com/en-us/security/business/security-101/what-is-cyber-threat-intelligence (nd)
17. MITRE: Att&ck threat report annotation model (tram): Automatically annotating cyber threat intelligence reports. https://github.com/mitre-attack/tram (2020)
18. with Natural Language Generation, A.A.C.T.I.R.: arxiv (2023), https://arxiv.org/abs/2310.02655
19. Networks, P.A.: Cyber threat intelligence and mitre att&ck framework. https://www.paloaltonetworks.com/resources/cyber-threat-intelligence-mitre-attack (2024)
20. Orbinato, V., Barbaraci, M., Natella, R., Cotroneo, D.: Automatic mapping of unstructured cyber threat intelligence: An experimental study (2022)
21. Powers, D.M.W.: Evaluation: From precision, recall and f-score to roc, informedness, markedness & correlation. Journal of Machine Learning Technologies **2**(1), 37–63 (2007). https://doi.org/10.1016/j.jmlr.2007.10.003
22. Rani, N., Saha, B., Maurya, V., Shukla, S.K.: Ttphunter: Automated extraction of actionable intelligence as ttps from narrative threat reports. In: ACSW '23: Proceedings of the 2023 Australasian Computer Science Week. pp. 126–134 (2023). https://doi.org/10.1145/3579375.3579391
23. Rani, N., Saha, B., Maurya, V., Shukla, S.K.: Ttpxhunter: Actionable threat intelligence extraction as ttps from finished cyber threat reports. arXiv (2024), https://arxiv.org/abs/2403.03267
24. Rastogi, N., Dutta, S., Christian, R., Gridley, J., Zaki, M., Gittens, A., Aggarwal, C.: Inferring cyber threat intelligence – a knowledge graph-based approach (2023)
25. Siracusano, G., Sanvito, D., Gonzalez, R., Srinivasan, M., Kamatchi, S., Takahashi, W., Kawakita, M., Kakumaru, T., Bifulco, R.: Time for action: Automated analysis of cyber threat intelligence in the wild. arXiv (2023), https://arxiv.org/abs/2307.10214
26. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management **42**(5), 1316–1342 (2006). https://doi.org/10.1016/j.ipm.2005.08.001
27. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management **42**(5), 1316–1342 (2016). https://doi.org/10.1016/j.ipm.2016.04.002