Noise Contrastive Estimation-based Matching Framework for Low-resource Security Attack Pattern Recognition

Anonymous ARR submission

Abstract

Tactics, Techniques and Procedures (TTPs) are standardized components in understanding cyber attack behaviours. Identifying TTPs in security writings, often called *TTP mapping*, is an important task to analyze and defend cybercrime. To tackle challenges of automated TTP mapping, conventional approaches resort to multi-class or multi-label learning. This setting suffers from a large number and complex *hierarchical* structure of the TTPs label space, as well as from the label imbalance. We propose a different learning paradigm, where TTPs labels are represented in the same space as cybersecurity texts, and the TTP mapping becomes ranking TTPs labels with highest semantic similarity scores to the input text. To this end, we propose a neural matching architecture with an effective sampling-based mechanism. Our experiments show that our method outperforms existing approaches when learning to match TTPs labels in the long tail. Furthermore, we make our annotated dataset public to foster further research in the field.

1 Introduction and Background

Cyber Threat Intelligence (CTI), an essential pillar of cybersecurity, involves collecting and analyzing information on cyber threats, including threat actors, their campaigns, and malware, helping timely counterintelligence and defending efforts. Textual threat reports or blogs are considered a crucial source of CTI, where security vendors diligently investigate and promptly detail intricate attacks. A key and intricate sub-task in extracting CTI from these textual sources involves the identification of Tactics, Techniques, and Procedures (TTP) of the threat actors, i.e. comprehending descriptions of low-level, complex threat actions and connecting them to standardized attack patterns. One of the popular standard knowledge frameworks widely adopted in the CTI community is MITRE ATT&CK (Storm et al., 2020). Within this framework, a technique represents a specific method used to achieve an objective, with its corresponding tactics and sub-techniques covering broader strategies and variations. Figure 1 illustrates an example of a text in a threat report, of which indicating two attack patterns, among others, i.e., (1) the use of a malicious email attachment to take control of a victim's system (T1566 [1]), and (2) encrypting data on the victim's system for ransom demands. (T1486 [2]).

[...] We witnessed that the botnet was spread via mass phishing, using a VB−scripted Excel attachment to download the second stage from xx.warez22.info. The same domain was used for C&C via HTTP. The botnet distributed a file encryption module we named VBenc. [...]

Figure 1: A fictional attack described in typical cybersecurity threat report writing style.

As of 2023, there are over 600 techniques, together with 14 high-level tactics described in MITRE ATT&CK. In its ontology, a technique is associated to at least one tactic (e.g., the technique "Hijack Execution Flow" is listed under three distinct tactics: Persistence, Privilege Escalation and Defense Evasion) and may have several sub-techniques. Mining techniques from CTI reports poses significant challenges due to several factors. Firstly, the large number of techniques, coupled with their diverse nature, intricate interdependencies, and hierarchical structure, renders the task complex and laborious. Secondly, the analysis of CTI reports necessitates the expertise of security professionals. The reports focus on delineating low-level threat actions rather than explicitly mentioning the associated techniques and tactics. Consequently, extracting relevant techniques and tactics from these reports requires diligent inference by the reader. Employing an automated

---

[1] `attack.mitre.org/techniques/T1566`
[2] `attack.mitre.org/techniques/T1486`

approach to TTP mapping presents inherent challenges. One major hurdle is the *low-resource* nature of the task, due to the limited availability of labeled data and the extensive label space. Moreover, the presence of long-tail infrequent TTPs adds complexity to the learning process.

Due to these challenges, TTP mapping has not been fully solved in related work. Most recent works use a classical *document*-level multi-label (Li et al., 2019) or *sentence*-level multi-class classification (Orbinato et al., 2022; You et al., 2022) learning setting. These granularity choices, however, either introduce unneeded complexity of long-form text representation (for *document*-level) or make the task inapplicable to mapping complex TTPs, which often require longer text (for *sentence*-level). Moreover, the main learning issues in these settings are: (i) the aforementioned problems of label scarcity and long-tailedness, and (ii) the learning complexity costs of the softmax-based learning approaches grow proportionally to the number of classes. In the wider literature i.e., extreme multi-label text classification (XMTC), the problems are addressed by (i) capturing the label correlation and (ii) partitioning and handling the sub-label spaces separately. They are, however, most effective in relatively resource-rich settings, and have drawbacks when applied to *label-scarce* scenarios, as the signal-to-noise ratio increases (Bamler and Mandt, 2020). In the multi-label context, learning is greatly affected, additionally, by the frequent presence of *missing* labels, which is a common trait observed in human-curated datasets.

In this work we propose an alternative learning setting which avoids the direct optimization for discriminating between data points in a large label space. Concretely, we transform the task into a *text matching* problem (Tay et al., 2018; Wang et al., 2017), allowing us to utilize the direct semantic similarity between the *input-label* pairs to derive a calibrated assignment score. The framework inherently incorporates an *inductive bias*, encouraging the capture of nuanced similarities even in the presence of limited labeled data, enhancing its ability to generalize to long-tail TTPs. This transformation is achieved by leveraging the *textual profile* of a TTP (i.e., textual *description* [3] in ATT&CK), a resource that is often neglected in related work.

Label-efficient text matching: Our approach -– dynamic *label-informed* text matching – exploits the shared information between a pair of texts (similar to text matching) in the learning phase, and altogether attempts to discriminate between the positive labels versus the rest in the label space (similar to classification). Powered by a sampling-based estimation method, i.e., Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010), this approach exhibits notable efficacy, in the moderately sized label space of TTPs, under data constraints.

To this end, we summarize our contributions:

- We formally redefine the challenging task of TTP mapping as a *paragraph-level hierarchical* multi-label text classification problem and propose a new learning paradigm that works effectively on the nature of the task.
- We introduce robust ranking-based NCE losses, designed not only to effectively handle the large label space but also the *scarce* and *missing* labels problem specific to this task. Additionally, we present a multi-task learning strategy that adeptly captures the intrinsic hierarchical structure within the label semantics.
- We curate and publicize an expert-annotated dataset that emphasizes on the multi-label nature, with approximately two times more labels per sample than existing datasets.
- Lastly, we conduct extensive experiments to prove our learning methods outperform strong baselines across real-world datasets.

2 Related Work

TTP Mapping and CTI Extraction Several works target TTP mapping on the *document level*. (Husari et al., 2017) used a probabilistic relevance framework (Okapi BM25) to quantify the similarity between *bag-of-words* representations of TTPs and the target text. However, this approach is limited to the oversimplified vocabulary of threat actions within an *ad-hoc* ontology. Ayoade et al. (2018); Niakanlahiji et al. (2018); Legoy et al. (2020) used a TF-IDF-based document representation and leveraged classical (i.e., tree-based, margin-based) ML for (multi-label) classification. Li et al. (2019) used latent semantic analysis to extract topics from target articles, and compared the topic vectors with the TF-IDF vectors of ATT&CK description pages to obtain cosine similarity. They used the similarity vectors with Naïve Bayes and decision trees to classify TTPs. However, the choice of document-level granularity introduces additional unneeded

---

[3] A technique, its description and procedure examples: `attack.mitre.org/techniques/T1021/`

complexity of long-form text representation. Recent works leverage transformers for *sentence-level* text representation learning (Orbinato et al., 2022; You et al., 2022), using the encoded representation in the multi-class classification setting. However, with limited available data, they restrict the task to only a small number of TTPs.

Extreme Multi-label Text Classification. XMTC, or generally extreme multi-label classification is a line of research targeting extremely large label spaces, e.g., product categorization in e-commerce or web page categorization. The main challenges for XMTC are computational efficiency and data skewness. Common techniques for XMTC are tree-based (You et al., 2019; Jasinska-Kobus et al., 2020; Wydmuch et al., 2018), sampling-based (Jiang et al., 2021) and embedding-based (Chang et al., 2021) that attempt to partition the label space and thus reduce the computational complexity. However, generally, these methods assume the sufficient availability of supervision and still suffer in the long-tail performance.

Matching Networks. Deep matching networks have witnessed rapid progress recently, finding applications in various conventional (e.g., retrieval (Wang et al., 2017)) or emerging tasks (e.g., few-shot (Vinyals et al., 2016) and self-supervised learning (Chen et al., 2020)). They can be architecturally categorized as *cross*- vs *dual*-encoder networks and can be optimized in tandem with the *triplet* (Schroff et al., 2015) or *contrastive loss* (Chopra et al., 2005). The former loss considers triplets of examples (anchor, positive, negative) and is *marginal*-based, whereas the latter, broadly referred to as NCE (Gutmann and Hyvärinen, 2010), utilizes a probabilistic interpretation. Despite demonstrating promising results across various domains and datasets, matching networks necessitate substantial training data. Although the NCE framework partially mitigates this concern, the well-adopted approach by Oord et al. (2018) remains somewhat limited, especially to the *fully-supervised* settings. Our approach overcomes the present constraints of training matching networks in settings where resources are limited, specifically when there is a scarcity of extensive training data.

3 Preliminaries and Problem Setup

In this section, we first provide a brief overview of the †classification settings with noise contrastive estimation (NCE). These definitions then subsequently help us in formulating our *matching* problem.

Classification: Let X and Y denote the *input* and *label* spaces, $|\mathsf{Y}| < \infty$. We define a score function $g_\theta : \mathsf{X} \to \mathsf{Y}$. In this setting, the *label* space Y is categorical. Specifically, $\mathsf{X} \in \mathbb{R}^{n \times m}$, whereas $\mathsf{Y} \in \{0,1\}^{n \times |L|}$, with n being the number of samples and L being the label set.

Matching: In this setting, X and Y represent the same *input* space. The matching function $g_\theta : \mathsf{X} \times \mathsf{Y} \to \mathbb{R}$, is differentiable in $\theta \in \mathbb{R}^{|\mathcal{D}|}$, where D is the parameter space. In order to cast a *classification* problem as a *matching* one, we assume there is an invertible and smooth *projection* function π that transforms the discrete categorical representation Y into the same continuous space as X.

Cross-entropy Loss and NCE: In either *classification* or *matching* settings, our goal is to estimate whether $\theta : x \mapsto \max_{y \in \mathsf{Y}} g_\theta(x,y)$ has optimal 0-1 loss. This can be reduced to conditional density estimation. Let $p_\theta(y|x) = \frac{\exp(g_\theta(x,y))}{\sum_{\hat{y} \in \mathsf{Y}} \exp(g_\theta(x,\hat{y}))}$, the cross-entropy loss is then defined as:

$$\mathsf{J}_{CE}(\theta) = \mathsf{E}_{(x,y)\sim(X \times Y)}[-\log p_\theta(y|x)] \qquad (1)$$

When Y is large, $\mathsf{J}_{CE}(\theta)$ is difficult to compute as the computation of the normalization term of $p_\theta(y|x)$ becomes expensive. This issue is addressed by NCE through sub-sampling $p(X,Y)$, and shifting the focus towards estimating the probabilities of the true data samples.

Multi-label Classification. The vanilla classification problem can be defined as follows: Let $\{X, Y\}$ be the problem space, where the feature space $\mathsf{X} \in \mathbb{R}^{n \times |\mathcal{D}|}$, and the label space $\mathsf{Y} \in \{0,1\}^{n \times |\mathsf{L}|}$, with $|\mathsf{L}| \ll \infty$ being the number of TTPs in the KB. The goal is to learn a function $f : \mathcal{D} \mapsto \mathbb{R}^{|\mathsf{L}|}$ that accurately predicts the multi-label one-hot vector output $y \in \mathsf{Y}$, given $x \in \mathsf{X}$.

Problem Reformulation. Given the training data $\mathsf{X} \in \mathbb{R}^{n \times |\mathcal{D}|}$, and $\mathsf{Y} \in \mathbb{R}^{|\mathsf{L}| \times |\mathcal{D}|}$, with $y \in \mathsf{Y}$ derived from the TTP *textual profile*, and $|\mathsf{L}| \ll \infty$ along with a set of supervisions $\{x \mapsto y\}^n = \{0,1\}^n$, such as $x \in \mathsf{X}$ and $y \in \mathsf{Y}$, our target is to learn *matching*-based scoring functions $\mathsf{g}_\theta(x,y)$ that model the relationship between x and y within the same feature space, aiming for $\mathsf{g}_\theta(x,y) \approx \{x \mapsto y\}^n$. The use of the *textual profile* inherently eliminates the need for an *projection* function π, as it directly aligns the discrete categorical representation Y with the same continuous space as X. In the context of cross-entropy loss, $p_\theta(y|x)$ is now linked to $p_\theta(x \mapsto y|x, y)$.

4 Methodology

Section describes our architectural choice for the matching function $\mathsf{g}_\theta(x, y)$, and our learning paradigm that approximates $p_\theta(x \mapsto y|x, y)$ to simultaneously match and compare TTPs labels.

4.1 Matching Network

The architecture of our matching network is built upon the *dual*-encoder framework, which typically employs a Siamese network. This shared network is used for learning the representations of both the target text segment and the TTP *textual profile*. As depicted in Figure 2, at a high level, our network comprises an embedding component and an alignment component. Each includes specific layers aimed at enhancing the connectivity between the two sub-network sides. Finally, the two sides are merged (by i.e., a dot product) to output a (probabilistic) *matching* score. We detail the architectural choice for our matching network below.

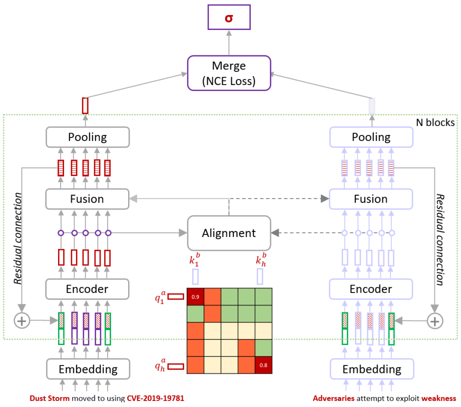

Figure 2: The dual-encoder matching network.

Encoder. The encoder has two modes: (1) *scratch* and (2) *scratch* with a pre-trained transformer (i.e., SecBERT) combined. Scratch indicates that the token embedding are learnt (with the embedding layer). We then apply a simple CNN on top of the embedding layer. With *scratch* alone, a specialized tokenizer (that respect CTI entities, e.g., URL, vulnerability identifier..) is used. While using together with the transformer, the tokenizer of the transformer is used. For (2), we simply stack the encoded vectors from the 2 sources together.

Alignment Network. Formally, given the input representation of the text-TTP pair as $x_t = (\hat{a}_1, \ldots, \hat{a}_l)$ and $y_{ttp} = (\hat{b}_1, \ldots, \hat{b}_l)$, the unnormalized attention weights are decomposed into: $e_{ij} = W^{align}(\hat{a}_i) \cdot W^{align}(\hat{b_j})$, whereas W^{align} is a trainable projection matrix, $\cdot$ is the dot product. Then, we derive the normalized weights for each token a_i and b_j, and achieve the corresponding alignment features.

Similar to (Yang et al., 2019), we use the block-based residual architecture with skip connections. Our block consists of the encoder, alignment and fusion layers. The fusion layer does various comparisons of local and aligned representations (i.e., the Hadamard product) and finally fuses the interaction vectors together using the concatenation operator. Then *pooling*, i.e., (non-) weighted average or max-pooling, is applied to attain fixed-length vector representations.

4.2 Learning

Our efficient learning method aims to circumvent the computational complexities that arise in the large label space, whether in the proper multi-label setting or its reduced multi-class version. The new learning paradigm is shifted from multi-label classification to the so-called *dynamic* label-informed text matching, in which negative labels are drawn dynamically at every step. The *ranker*, acting as a simultaneous matcher, strategically assigns higher probabilities to positive pairs and lower probabilities to negative pairs. Finally, the top-k positive pairs are selected based on a cut-off threshold. We detail our learning mechanism below.

***Partial-ranking*-based NCE**. The general idea of NCE in our scenario is to avoid an exhaustive ranking (or partitioning) in the large label space, i.e., in the vanilla multi-label classification setting. Instead, a matching-based classifier, $p((x \mapsto y)|x, y)$, is trained to differentiate between samples from the true distribution and a noise distribution, $q(y)$, and inherently approximate the underlying ranking function. By utilizing Monte Carlo sampling, the NCE loss can be formulated as follows:

$$\begin{aligned} \mathsf{J}_{NCE}(\theta) = \mathsf{E}_{(x,y)\sim(\mathsf{X}\times\mathsf{Y})}(\log p((x \mapsto y) = 1|x, y) \\ + \sum_{i=1, y_i \sim q}^{k} \log p((x \mapsto y) = 0|x, y_i)) \end{aligned} . \qquad (2)$$

While the NCE loss in Equation 2 is calculated by learning $p((x \mapsto y)|x, y)$ for every data point (so-called *local*), we opt for a *ranking* setting where data points in the same batch *compete* in a contrastive setting. One way of achieving this is to use the mutual information, as utilized in InfoNCE (Oord et al., 2018), to quantify the distance between the prediction and (k-sized) label distributions. The ranking NCE loss is then defined as:

$$\mathsf{J}_{NCE}^{global} = -\mathsf{E}_{(x,y)}[\log \frac{\exp(g_\theta(x_i, y))}{\gamma \sum_{j:(x \mapsto y_j)=0} \exp(g_\theta(x, y_j))}] , \quad (3)$$

whereas, $g_\theta(x, y)$ is the *matching* function. Consequently, minimizing the loss promotes simultaneously a lower g_θ for negative pairs and a higher g_θ for positive pairs. The scaling factor γ, which is absent in InfoNCE, is introduced to account for the need to reduce the impact of the considerably larger portion of negative samples. This adjustment aims to emphasize the top-k *partial* ranking, where it is assumed that the positive samples are concentrated in the distribution. Subsequently, when γ presents, the loss is denoted as **α-balanced** NCE.

Asymmetric Focusing. Given the limited availability of reliable labels, our objective is to (i) reduce the impact of straightforward negative samples, and (ii) simultaneously mitigating the influence of potentially *mislabeled* (due to *missing* or *wrong* labels) samples on the loss function. While (i) can be achieved by applying a (hard) cut-off on very low values of $p(0|x, y_i)$, (ii) is often attributed to the high $p(1|x, y_i)$, with $y_i \sim q$. Thus, we opt for an *asymmetric* approach for the design of the NCE loss, wherein we prioritize the challenging mislabeled samples. In doing so, we explicitly differentiate the focusing (scaling) levels between the positive and negative groups. The idea originated in Ridnik et al. (2021), for vanilla cross-entropy. In our case, the negative samples derived from our negative sampling strategy in the NCE context. Our hypothesis is that this asymmetric mechanism helps stabilize the learning towards the *noisy*[4] sampled negative labels. Let γ^+ and γ^- be the positive and negative scaling parameters, respectively. The sample-level *asymmetric* loss is achieved as follows:

$$\begin{aligned} \mathsf{J}^{(+)} &= (1-p)^{\gamma_+} \log(p); \\ \mathsf{J}^{(-)} &= p^{\gamma_-} \log(1-p), \end{aligned} \quad (4)$$

where γ_- is often set larger than γ_+ and p is short for $p((x \mapsto y)|x, y)$. The NCE loss is obtained by aggregating J over all samples.

$$\mathsf{J}_{NCE} = J^{(+)}(x, y) + \sum_{i=1, y_i \sim q}^{k} \mathsf{J}^{(-)}(x, y_i). \quad (5)$$

To this end, we show in Algorithm 1 our NCE-based training procedure. The convergence analysis can be further found in Appendix B.

Algorithm 1 *NCE-based training procedure*

```
Input: Parameters $\theta$, learning rate $\epsilon$.
Empirical data distribution $\hat{p}_d = (x_i, y_i)_{i=1}^{n}$
for each epoch do
    for t=1,2.. do
        Sample $i, i'_k \sim [1, .., n], k \in [1, .., K]$
        $g_{(+)} = g_\theta(x_i, y_i)$
        $g_{(-)} = g_\theta(x_i, y_{i'_k})$
        logits $= \{g_{(+)}, g_{(-)}\}$, labels $= \{0, 1\}$
        # compute $\alpha$-balanced or asymmetric loss
        $\mathcal{J}_{NCE} = \log \sum_k (exp(g_\theta(x_i, y_{i'_k}) - \gamma \cdot g_\theta(x_i, y_i))$
        # use SGD optimizer
        $G^{(t)} \leftarrow G^{(t)} + \frac{1}{m} \nabla_\theta \mathcal{J}_{NCE}(g_\theta)$
        $\theta \leftarrow \theta + \epsilon\, cdot G^{(t)}$
    end for
end for
```

4.3 Sampling Strategies

Corpus-level negative sampling. Due to memory constraints, the conventional negative sampling method is often applied *in-batch* (Yih et al., 2011; Gillick et al., 2019). However, as the sampling scope is restricted to be at the batch level, the negative sampling (noise) distribution estimated from in-batch sampling may be biased or incomplete. As the size of our label collection is not as large as in related tasks, the *corpus-level* sampling strategy is more appropriate. In effect, we assume that a larger part of the TTP corpus is *irrelevant* to the positive paired sample. We also assume that noisy samples will inherently be canceled out while learning signals remain in our training paradigm (Rolnick et al., 2017). While being simple, the policy *augments* our dataset with a substantial supervision signal stemming from negative samples. We explain the details of our sampling policies below.

Random sampling. A simple uniform distribution $q(y) = \frac{1}{\|L\|}$ or the population marginal distribution $q(y) = p_{\mathsf{X,Y}}(y)$, or an empirical approximation of it. We opt for the former when refering to the technique henceforth. To increase the hardness of negative samples, other sampling methods, i.e., retrieval-based (e.g., candidates from a retrieval model) or semantic structure-based (e.g., other sibling TTPs of the same technique) can be applied. However, due to the missing label nature of the task, these hard techniques tend to introduce noisy bias and thus are sub-optimal.

4.4 Hierarchical Multi-label Learning

In ATT&CK, TTPs have a hierarchical structure, where different sub-techniques map *many-to-1* to

---

[4] Which *negative* samples are not exclusively negative?

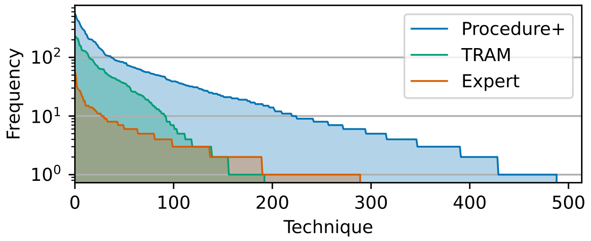

Figure 3: The distributions of the number of samples per technique (TTP) for each dataset.

the same technique and techniques map *many-to-many* to tactics. To exploit and encode this structure, we design an *auxiliary* task that predicts the tactics of the textual input, alongside our *matching* task. This auxiliary task is thus also a medium-sized multi-label classification task, and we use the binary cross-entropy loss for the optimization. The two tasks are jointly optimized in a *multi-task* learning manner, where the two losses are linearly combined: $\mathrm{J}_{total} = \alpha \mathrm{J}_{NCE} + \beta \mathrm{J}_{aux}$, where α and β are loss-weighting parameters.

5 Experiments

5.1 Datasets

We list below the datasets used in our experiments.

TRAM. Largest publicly available manual curated dataset from CTID [5], commonly used in related work. It comprises mostly short texts, covers only one-third of TTPs with relatively noisy labels, thus appears to have limited application value.

Procedure+.[6] *Procedures*: collected from ATT&CK, where techniques have associated manually curated procedure examples [3]. Each example is a one-sentence expert-written summary of the implementation of a technique in real-world attacks. *Derived Procedures*: complements an example with a text that aligns to threat report writing style. We look for evidential paragraphs in the references where the summary example is assumedly derived from, using a per-document search engine.

Expert.[6] Our purposefully crafted dataset closely emulates real-world scenarios, providing an practical setting for TTP extraction. Unlike sentence-focused datasets, ours covers entire paragraphs, thus the annotations are inherently multi-label in nature. Annotated by 5 CTI experts using an in-house tool, our dataset triples text length and increases average labels per sample by approximately 60-80% compared to TRAM (see Table1).

Table 1: Dataset statistics. *S+T* denotes the joint count of techniques and sub-techniques.

Dataset	Texts	S+T	Tech-niques	Avg. # Labels	Avg. # Tokens
TRAM	4797	193	132	1.16	23
Procedures	11723	488	180	1.00	12
Derived Procedures	3519	374	167	1.22	65
Expert	695	290	151	**1.84**	**72**

In our experiments, the two procedure examples datasets serve as high-quality *pseudo*-datasets, providing additional training examples, as well as valuable benchmarking perspective. Further descriptions of the overall dataset construction processes can be found in Appendix C.

5.2 Metrics and Baselines

The following common metrics in literature are used: the micro-averaged $\{\mathbf{P}, \mathbf{R}, \mathbf{F1}\}@k$ and mean reciprocal rank (**MRR**)$@k$, which measures the relative ordering of a ranked list.

The following baselines are targeted: **Okapi BM25**, adjusted from Husari et al. (2017). The *bag-of-word*s model is enhanced by a security GloVe LM and TTP textual profile are used.

Binary Relevance, the vanilla multi-label learning approach, similar to Li et al. (2019) for TTP mapping. It has the one side of the text matching architecture and learns a binary classifier for each label separately in a one-vs-all manner.

Dynamic *triplet*-loss, a competitive baseline with a similar network architecture to ours, employs a *triplet*-based loss (Schroff et al., 2015). In contrast to the (empirically found) ineffective vanilla setting, we dynamically generate k-negative samples (akin to N-pairs loss (Sohn, 2016)) to mimic the NCE mechanism.

In addition, we employ the following state-of-the-art (SoTA) models in XMTC as competitive baselines: **NAPKINXC (Jasinska-Kobus et al., 2020)**, a method that generalized the Hierarchical Softmax, so-called Probabilistic Label Trees (PLT), commonly used in XMTC literature. **XR-LINEAR (Yu et al., 2022)**, a model designed for very large output spaces, with 3 phases: semantic label indexing (label clustering), matching (where the most relevant clusters are identified), and ranking (of labels in the matched clusters). **XR-TRANSFORMER (Zhang et al., 2021)**, similar

---

[5] CTID TRAM: `github.com/CTID/TRAM`

[6] The datasets will be publicly shared at `http://anonymized-url` to foster further research.

Table 2: Results of all models on 3 datasets. *Procedures*+ denotes the combined procedure examples datasets. Bold denotes *best* while underscore signifies *second-best* performance. Indented $(-)$ denotes training **without** the specific option wrt. the preceding model. *Ideal* R@1 on the Expert dataset is 0.504. $^{\mathcal{T}}$ uses pre-trained SecBERT.

		Procedures+				**TRAM**				**Expert**			
	Methods	P@1	R@1	F1@3	MRR@3	P@1	R@1	F1@3	MRR@3	P@1	R@1	F1@3	MRR@3
Baseline	TTPDrill (BM25)	.230	.227	.118	.232	.250	.212	.118	.205	.222	.037	.008	.139
	Binary Relevance$^{\mathcal{T}}$	.206	.579	.193	.579	.236	.594	.209	.594	.189	**.256**	.085	.256
	Dynamic Triplet-loss$^{\mathcal{T}}$	.339	.336	.277	.432	.286	.253	.277	. 402	.449	.111	.252	.525
XMTC	eXtremeText (Sigmoid)	.557	.547	.371	.624	.632	.594	.425	.729	.407	.174	.279	.485
	eXtremeText (PLT)	.528	.519	.336	.582	.612	.578	.393	.671	.344	.146	.243	.411
	NAPKINXC	.578	.570	.383	.661	.662	.614	.453	.754	.497	.199	.365	.582
	XR-LINEAR	.604	.595	.393	.684	.674	.626	.445	.757	.529	.215	.363	.600
	XR-TRANSFORMER$^{\mathcal{T}}$	.502	.494	.304	.548	.540	.515	.334	.595	.389	.149	.239	.453
Ours	InfoNCE$^{\mathcal{T}}$	.672	.639	.442	.758	.697	.577	.516	.799	.702	.175	.432	.768
	@−balanced$^{\mathcal{T}}$	**.760**	**.720**	.489	.837	.765	.646	.546	.856	.693	.169	.400	.762
	(−) auxiliary	.604	.584	.433	.719	.712	.601	.521	.816	.693	.177	**.442**	.773
	(−) Transformers	.646	.601	.357	.772	.642	.543	.547	.785	.700	.173	.430	.766
	Asymmetric$^{\mathcal{T}}$	.757	.718	**.493**	**.838**	**.770**	**.658**	**.555**	**.864**	**.731**	.182	.399	**.789**

to XR-LINEAR, but with a transformer encoder. **exTremeText (Wydmuch et al., 2018)**, algorithm-wise relatively similar to NAPKINXC.

5.3 Experimental Setup

We use the common security LM SecBERT[7] for the transformer-based models, and grid search determined the best hyperparameters for each model. The rich textual description [3] of a TTP is selected for the textual profile. Except for XMTCs and BM25, all models are with the *auxiliary* tasks.

Data Settings. For the *Procedure*+ and TRAM datasets, each was *stratified*-shuffled and split into training, validation and test sets with ratios of 72.5%, 12.5% and 15%, respectively. The test sets remained fixed for reporting purposes. For training and validation, two modes were considered: *separate* and *combined*. In the former, the datasets are kept distinct, while in the latter, they were merged according to their respective splits.

For the Expert dataset, we utilize a dedicated *held-out* recall-focused test set, with 157 unique paragraph-level samples and 3.3 labels per sample on average. This carefully curated held-out set closely resembles paragraph-level text snippets in complete CTI reports, facilitating a comprehensive analysis of the entire report.

5.4 Results and Analysis

Table 2 presents the main experimental results. Overall, our proposed NCE-based models greatly outperform the baselines. Particularly, the *asymmetric* loss-based model achieves the best performance across most metrics and datasets. We also observe the significant improvements of the two loss variants (i.e., α-balanced and *asymmetric*) over the vanilla InfoNCE. In addition, the models demonstrates a substantial improvement at the cutoff threshold @1 ($\sim$10%) in comparison to @3 ($\sim$5%). This supports the effectiveness of our *matching* network in *classification* settings.

The SoTA XMTC baselines perform considerably robust across the three datasets, among these XR-LINEAR perform best. Interestingly, XR-LINEAR demonstrates consistently higher performance than its related transformer-based counterpart (XR-TRANSFORMER), suggesting the challenges of the larger models in our low-resource settings. We also observe the subpar performance of the *triplet*-loss approach, suggesting similar disadvantages in the low-resource settings.

Across the datasets, the overall model performance declines from Procedure+ to TRAM and Expert, indicating varying complexities within each dataset. Notably, our performance yields compelling results in TRAM, well-surpassing methods commonly reported in related work, i.e., BM25 and Binary Relevance.

5.5 Ablation Studies

Hierarchical Labeling. We analyze the contributions of our *hierarchical* modeling to the ranking performances. As shown in Table 2, in general, our joint learning with the *auxiliary* task gives a notable performance boost in most scenarios. Further analysis can be found in Appendix D.3.

---

[7] `https://github.com/jackaduma/SecBERT`

Table 3: Model performance on the *head* vs. *tail* parts of the TRAM dataset. *Head* denotes more frequent TTPs ($>$ *empirical* **7** samples in the *training* split), whereas *tail* are infrequent TTPs. All are trained in *combined* mode. **Bold** denotes *absolute* best performers.

	TRAM *head (94.5%)*			**TRAM** *tail (5.5%)*		
Methods	F1@1	F1@3	MRR@3	F1@1	F1@3	MRR@3
BM25	.195	.112	.21	+118%	+99.1%	+108%
NAPKINXC	.624	.458	.752	-36.9%	-27.1%	-30.2%
XR-LINEAR	.62	.448	.743	-16.3%	-25.4%	-21.5%
@-balanced	.668	**.548**	.841	**-3.3%**	**-12.2%**	**-8%**
Asymmetric	**.679**	.547	**.848**	-4.9%	-14.3%	-10.4%

Table 4: Model performance on the *head* vs. *tail* parts of the Expert dataset. Legend of Table 3 applies.

	Expert *head (56.5%)*			**Expert** *tail (43.5%)*		
Methods	F1@1	F1@3	MRR@3	F1@1	F1@3	MRR@3
BM25	.071	.107	.188	+26%	+28%	+18.6%
NAPKINXC	.334	.381	.655	-40.7%	-23.9%	-16.6%
XR-LINEAR	**.335**	.407	.676	-31.6%	-22.9%	-14.5%
@-balanced	.302	**.426**	.819	-18.2%	**-11.3%**	-2.9%
Asymmetric	.306	.416	**.831**	**-18.9%**	-12%	**-2.9%**

Transformers. We observe the positive contributions of SecBERT to the performance of all models in most cases. Nevertheless, without SecBERT (i.e., (−) Transformers), our models still very much on par with the strong XMTC baselines at $k = 1$ and outperform them at $k = 3$, indicating the better ranking capability, specially on the *Expert* dataset.

Long Tail Analysis. Tables 3 and 4 provide an analysis on the models' performances on the classes of *head* versus *tail* frequency distributions visualized in Fig. 3. Overall, *matching*-based approaches, with the inductive bias, are relatively robust, whereas the classification-based XMTC baselines suffer in the long tail.

Loss Analysis. In Figure 4, we present additional analysis on the impact of the *size* of negative samples. The results indicate that as the size increases, the model tends to converge faster and exhibit better performance. However, it appears that there are no additional benefits beyond a size of 60, which corresponds to 10% of the label space.

A further analysis on the score distribution of the ranked lists are reported in Figure 5. The details are provided in the caption for convenient reference.

6 Conclusion

We proposed a solution for the TTP mapping task that overcomes low-resource challenges in security domain. This new learning paradigm integrates the inductive bias into the classification task, resulting in significant out-performance of strong baselines.

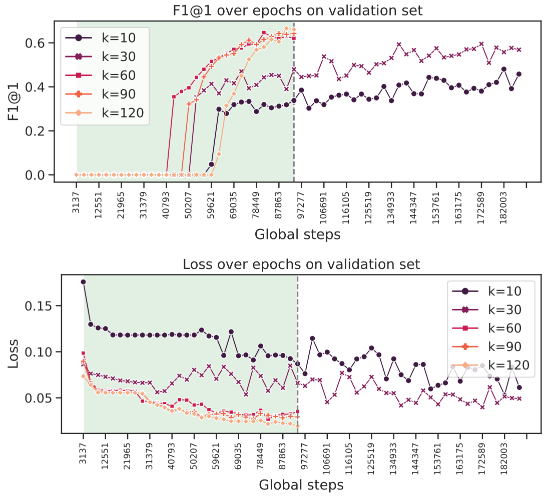

Figure 4: InfoNCE loss and f1@1 performance wrt. different number of negative samples. The network is without transformers. OOM for larger number of negative samples on an NVIDIA V100 32GB RAM.

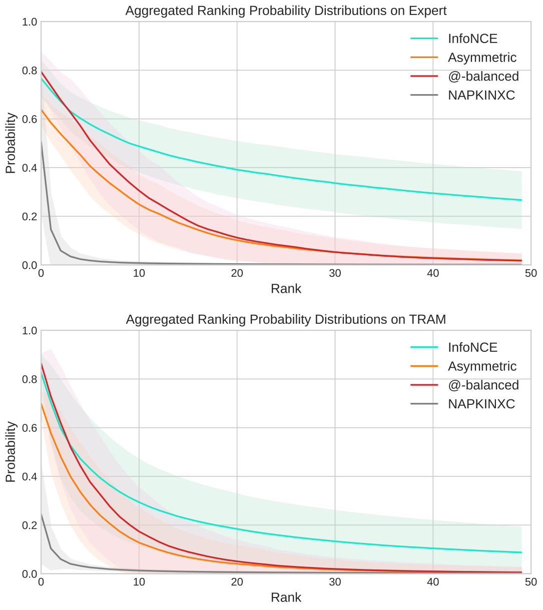

Figure 5: The aggregated probability distribution of the top-50 ranking on different models on the test splits of the TRAM (left) and Expert (right) datasets. While InfoNCE tends to allocate probabilities to labels in the long tail, @-balanced and *asymmetric* exhibit a more pronounced skewness in their distribution, resembling that of a pure classification model like NAPKINXC. The NCE-based models display a broader distribution at the head, indicating their inclination to assign comparable probabilities to multiple labels.

7 Limitations

Despite its label efficiency, our learning approach is not particularly efficient in terms of training. On average, it requires 24 hours for training on a machine equipped with a single NVIDIA-Tesla-V100 32 GB. Nonetheless, its training time is nearly comparable to the baselines employing Transformers. Although our expert dataset closely aligns with the multi-label nature of the task and exhibits higher quality, it remains relatively limited in size, covering just one-third of the TTPs.

8 Ethics Statement

Our datasets are constructed from security threat reports published by security vendors, and copyrighted by their respective owners. We scraped and extracted textual contents from these public websites to build the datasets. The criteria for text selection was whether the text discusses TTPs.

Some source reports contain Personally Identifiable Information (PII) of report authors, threat actors (i.e., persons suspected of involvement in cybercrime) or victims (i.e., persons suspected of being targeted by cybercrime). In the text selection process, we screened for any PII and removed all uncovered instances. However, we cannot rule out the possibility that some PII might have been missed in that process. Thus, users wishing to use the data will need to accept our terms of use and report potential remaining instances of PII, which will be removed in a subsequent dataset update. Crucially, the potential remaining PII in the dataset has been originally published by the reports' authors and may still remain public on the original websites even after our dataset updates.

The datasets have been annotated by security experts in our organization as part of their regular work under full-time employment contracts.

The language of the dataset is English, written by native and non-native speakers.

We are not aware of any ethical implications stemming from the intended use of this dataset, i.e., TTP mapping.

References

Gbadebo Ayoade, Swarup Chandra, Latifur Khan, Kevin Hamlen, and Bhavani Thuraisingham. 2018. Automated threat report classification over multi-source data. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 236–245. IEEE.

Robert Bamler and Stephan Mandt. 2020. Extreme classification via adversarial softmax approximation. (arXiv:2002.06298). ArXiv:2002.06298 [cs, stat].

Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. 2021. Extreme multi-label learning for semantic matching in product search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2643–2651.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, page 528–537, Hong Kong, China. Association for Computational Linguistics.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, page 297–304. JMLR Workshop and Conference Proceedings.

Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, page 103–115, Orlando FL USA. ACM.

Kalina Jasinska-Kobus, Marek Wydmuch, Krzysztof Dembczynski, Mikhail Kuznetsov, and Robert Busa-Fekete. 2020. Probabilistic label trees for extreme multi-label classification. *arXiv preprint arXiv:2009.11218*.

Ting Jiang, Deqing Wang, Leilei Sun, Huayi Yang, Zhengyang Zhao, and Fuzhen Zhuang. 2021. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7987–7994.

Andrej Karpathy. 2023. State of gpt.

Valentine Legoy, Marco Caselli, Christin Seifert, and Andreas Peter. 2020. Automated retrieval of att&ck tactics and techniques for cyber threat reports. *arXiv:2004.14322 [cs]*. ArXiv: 2004.14322.

Mengming Li, Rongfeng Zheng, Liang Liu, and Pin Yang. 2019. Extraction of threat actions from threat-related articles using multi-label machine learning classification method. In *2019 2nd International Conference on Safety Produce Informatization (IICSPI)*, page 428–431.

Amirreza Niakanlahiji, Jinpeng Wei, and Bei-Tseng Chu. 2018. A natural language processing based trend analysis of advanced persistent threat techniques. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2995–3000. IEEE.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

V. Orbinato, M. Barbaraci, R. Natella, and D. Cotroneo. 2022. Automatic mapping of unstructured cyber threat intelligence: An experimental study: (practical experience report). In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pages 181–192, Los Alamitos, CA, USA. IEEE Computer Society.

Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. 2021. Asymmetric loss for multi-label classification. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, page 82–91, Montreal, QC, Canada. IEEE.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29.

Blake E. Storm, Andy Applebaum, Doug P. Miller, Kathryn C. Nickels, Adam G. Pennington, and Cody B. Thomas. 2020. MITRE ATT&CK®: Design and Philosophy. Technical report, MITRE Corporation, McLean, VA.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Co-stack residual affinity networks with multi-level attention refinement for matching text sequences. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4492–4502.

Sun Tzu. *The Art of War*. 5th century BC.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems*, 29.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150.

Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. 2018. A no-regret generalization of hierarchical softmax to extreme multi-label classification. *Advances in neural information processing systems*, 31.

Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen. 2019. Simple and effective text matching with richer alignment features. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4699–4709.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, page 247–256, Portland, Oregon, USA. Association for Computational Linguistics.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 32.

Yizhe You, Jun Jiang, Zhengwei Jiang, Peian Yang, Baoxu Liu, Huamin Feng, Xuren Wang, and Ning Li. 2022. Tim: threat context-enhanced ttp intelligence mining on unstructured threat data. *Cybersecurity*, 5(1):3.

Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S Dhillon. 2022. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 23:1–32.

Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit Dhillon. 2021. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 34:7267–7280.

A The Task of TTP Mapping

In the cybersecurity domain, one of the pillars of effective defense is *Cyber Threat Intelligence* (CTI).

> *[...] We witnessed that the botnet was spread via mass phishing, using a VB−scripted Excel attachment to download the second stage from xx.warez22.info. The same domain was used for C&C via HTTP. The botnet distributed a file encryption module we named VBenc. [...]*

Figure 6: A fictional attack described in typical cybersecurity threat report writing style.

An analog to military intelligence, CTI is tasked with collecting and organizing information on cyber threats such as *threat actors*, their threat *campaigns*, and malicious software, i.e., *malware*. It can be traced back to ancient military-theoretical observations that understanding one's enemy is crucial to winning battles[8].

CTI describes cyber threats on three levels. The *strategic level* (e.g., periodicals on trends in the cyber risk landscape) describes high-level threat information and targets non-technical chief executives. The *tactical level* (e.g., technical reports on individual threat actors) describes details on threat actors' behavior, for use by security managers. The lowest, *operational level* (e.g., lists of malicious internet domains) describes specific threat indicators which may be directly used for defense (e.g., by blocking the offending domains).

While the value of CTI data is roughly proportional to its intelligence level, the difficulty of obtaining it is the opposite. Automated production only exists for operational CTI data, and higher levels require costly manual expert work. However, leading CTI community members regularly publish tactical and strategic CTI information in form of *cybersecurity threat reports* – digital documents with unstructured natural language text along tables and images, written using a domain-specific vocabulary, between hundreds and thousands of words long, and strongly interspersed with technical tokens such as URLs, hashes and similar. Topically they cover profiles of major threat actors, summaries of threat campaigns, and malware analysis reports. An illustrative excerpt is provided in Fig. 6. Thus an opportunity arose for a fruitful application of NLP: automated extraction of high-value CTI data from natural language documents.

In recent years, the NLP and cybersecurity communities have been engaged in exactly this direction. Early work targeted the operational level, extracting *Indicators of Compromise* (IoCs), i.e., threat actor controlled internet domains, IP addresses, file hashes and URLs, from security articles, social media or forum posts. Subsequent efforts targeted the tactical level, but the challenge there remains unsolved.

The tactical level characterizes adversaries' behavior, typically referred to as *attack patterns*. Fig. 6 illustrates, among others, (1) the use of a malicious email attachment to take control of a victim's system, and (2) encrypting data on the victim's system to extort money from the victim. To facilitate reasoning about attack patterns, of which hundreds are documented, the community converged around a common framework called *Tactics, Techniques and Procedures* (TTPs):

- A **tactic** describes the purpose of the actor's behavior – “why?”. For above examples, the tactics are *taking control of the system* and *financial gain*, respectively. Other typical adversarial tactics include *reconnaissance*, *establishing permanent presence*, *command and control*, *data theft*, etc.

- A **technique** describes the method used for the given purpose – “how?”. In our case, those are *malicious email attachment* and *data encryption*. A technique may be assigned to several tactics if it achieves several purposes. Each tactic can be achieved using any of a range of different techniques. Other typical techniques include *collecting victim system information*, *execution on system start*, *encrypted communication*, *password theft*, etc.

- Some ontologies also define a **subtechnique** as a specialized technique. A technique may be specialized by zero or more subtechniques. For example, the technique *input capture* may have subtechniques *keystroke capture* and *screen capture*.

- A **procedure** describes the implementation details of a technique. For example, the email attachment may be a *malicious Excel file*, and the data encryption may be performed using *a custom encryption algorithm*. Each technique can be implemented using any of potentially many different procedures.

---

[8] “If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.” (Tzu)

Although others exist, MITRE ATT&CK[9] (Storm et al., 2020) is the prevalent knowledge base and taxonomy of TTPs used in the literature. It currently comprises 14 tactics, 196 techniques, 411 subtechniques and thousands of procedures, continually curated by community experts.

Retrieval of TTPs from unstructured text is referred to as *TTP mapping* in this work, although *TTP mining/extraction* also occur in the literature. Crucially for TTP mining, threat reports very rarely name actors' TTPs explicitly. Instead, they establish a chronological narrative in terms of *threat actions*, i.e., low-level actions taken by the threat actor. Some examples for threat actions from Fig. 6 are *botnet spreading*, *use of phishing emails*, *use of Visual Basic for malicious scripting*, *use of Excel macros*, etc. Not all threat actions are explicitly expressed in the text. For example, although the term "email" is not mentioned, the use of phishing emails is inferred by domain experts because phishing means sending deceptive emails with malicious purposes, therefore sending emails is the technical implementation of phishing and it must have occurred.

Thus, at a high level, TTP mapping from text is a 3-step process:

1. Identification of individual threat actions from paragraphs or longer context
2. Correlation of one or more identified threat actions into procedures
3. Mapping of identified procedures into techniques and tactics.

B Convergence Analysis

Based on the stability of the NCE losses, we briefly discuss the convergence properties of our adjusted losses.

Boundedness of Gradients. *Proof*: Let $g(x, y)$ be the matching function such that $0 \leq g(x, y) \leq 1$ for all (x, y). Consider the NCE loss, i.e., @-balanced with a scaling factor γ:

$$J_{\text{NCE}}(\theta) = E_{p(x,y)}[\log g(x, y)] - \gamma E_{p(x)}[\log \textstyle\sum_j g(x, y_j)]$$

We want to prove that the gradients of the NCE loss with respect to the model parameters are bounded. Let $\nabla J_{\text{NCE}}(\theta)$ denote the gradient vector. Taking the partial derivative of $J_{\text{NCE}}(\theta)$ with respect to a parameter θ_i, we have:

$$\frac{\partial J_{\text{NCE}}(\theta)}{\partial \theta_i} = \frac{\partial}{\partial \theta_i}\left(E_{p(x,y)}[\log g(x, y)] - \gamma E_{p(x)}[\log \textstyle\sum_j g(x, y_j)]\right)$$

Using the linearity of the derivative, we can rewrite the above expression as:

$$\begin{aligned}\frac{\partial J_{\text{NCE}}(\theta)}{\partial \theta_i} &= E_{p(x,y)}\left[\frac{\partial}{\partial \theta_i}\log g(x, y)\right] \\ &\quad -\gamma E_{p(x)}\left[\frac{\partial}{\partial \theta_i}\log \sum_j g(x, y_j)\right]\end{aligned}$$

Since $0 \leq g(x, y) \leq 1$, the derivative of $\log g(x, y)$ with respect to any parameter θ_i is bounded between 0 and 1. Similarly, the derivative of $\log \sum_j g(x, y_j)$ with respect to θ_i can be bounded by considering the partial derivatives of $g(x, y_j)$.

Therefore, we can conclude that:

$$\left|\frac{\partial J_{\text{NCE}}(\theta)}{\partial \theta_i}\right| \leq \max\{1, \gamma \max_{x,y_j} |\partial_{\theta_i} g(x, y_j)|\}$$

The above inequality implies that the absolute value of the partial derivative of the NCE loss with respect to any model parameter is bounded by a finite value, scaled by γ. Hence, we have shown that the gradients of @-balanced with the scaling factor γ are bounded. The proof for the *asymmetric* loss can be derived in an analogous manner.

Lemma 1 *The matching function* $g(z_i, z_j)$ *is Lipschitz-continuous with a constant* C*, meaning that for any* z_i, z_i', z_j*, we have* $|g(z_i, z_j) - g(z_i', z_j)| \leq C|z_i - z_i'|$.

Informal proof. Our Siamese neural networks-based matching function $g(z_i, z_j) \in [0, 1]$. □.

Lemma 2 *The noise distribution q satisfies the matching moment condition of the true distribution p, which, in essence, indicates that the covariance matrices of the two are similar.*

Informal proof. Since the noise distribution is sampled over the whole corpus, the lemma holds true for the random sampling strategy. □.

Thus, our loss is also Lipschitz-continuous and retains convergence properties of the original NCE losses, when optimized using SGD together with the random negative sampling.

---

[9] `https://attack.mitre.org/`

C Dataset Construction

Derived Procedure Examples. The dataset is created as a contextualized version of the original *Procedure* examples. We search for evidential *paragraph-level* text snippets in the references where the summary example is derived from. With this, the examples are contextualized and reflect the true reporting style present in the references. The pre-processing steps are as follows:

- Each example-reference pair is indexed at the *paragraph* level. Any paragraphs that are deemed (1) too short (less than 20 tokens), (2) too long (more than 300 tokens), or (3) have a Jaccard index with the example exceeding 0.9 (indicating near-*duplicate*) are discarded.
- The remaining paragraphs are ranked based on their relevance to the example using a tailored BM25 retrieval model.
- A maximum of two paragraphs that satisfy a carefully chosen global cut-off threshold are selected.
- Additionally, we eliminate any potential near-duplicates to the TRAM and Expert datasets.

We further assessed the dataset quality on a limited sample set consisting of 50 text snippets. Through this qualitative evaluation, the overall impression of the examined samples is largely positive.

Expert Dataset.

The Expert dataset comprises relevant text paragraphs from articles of reputable cybersecurity threat researchers, annotated by seasoned cybersecurity experts. The dataset was purposefully designed to closely mimic real-world scenarios, aiming to provide a practical and authentic setting for TTP extraction. Unlike datasets that primarily focus on individual sentences, our dataset encompasses entire paragraphs, and the annotations are inherently multi-label in nature. Rather than concentrating on isolated sentences, this dataset includes entire paragraphs that contain implicit mentions of TTPs, making the annotations inherently multi-label in nature.

The dataset was collected as follows:

1. We scraped 30 thousand articles from the feeds of leading cyber threat research organizations, and heuristically filtered out irrelevant articles, which do not describe attacks related to malware, advanced persistent threats, or cyber threat campaigns.
2. Further heuristics were applied to remove irrelevant paragraphs, i.e., we look for paragraphs which satisfy aforementioned length constraints, and contain at least 3 cybersecurity entities (e.g., malware, URL, etc.). The remaining relevant paragraphs were then randomly sub-sampled for annotation.
3. The expert annotators were tasked with analyzing the paragraph and identifying TTPs. To assist them in this process, an in-house search engine, powered by the baseline retrieval model BM25, was employed. This search engine allowed the annotators to formulate queries based on the paragraph and retrieve relevant information to aid in their TTP selection.
4. The annotators were instructed to only annotate explicit tactics and techniques in the given paragraph[10].

Each annotated item, namely a text paragraph, undergoes evaluation by a single annotator. We refrained from implementing extra quality control procedures, such as reviews or reaching consensus among annotators. To ensure quality, we engaged seasoned cybersecurity experts as annotators, rather than relying on crowd-sourced workers.

The choice of text paragraphs is biased by the described selection process towards high-quality writing from expert threat reports, and might not be representative of other writing styles, e.g., microblogging posts.

Expert Dataset: Special Test Split. In the aforementioned process, it cannot be guaranteed that all annotations will be retrieved accurately due to the extensive task of re-formulating queries and reviewing the lengthy ranked list of TTPs generated by the relatively lower-performing BM25 model. Therefore, in order to enhance the recall of the test split, we substituted BM25 with our *InfoNCE* model, which was trained on the train splits of the *Procedure+* and Tram datasets. For every sample, we utilize a deep cut-off approach by selecting the top

---

[10] An expert may comprehend from the text that it would be impossible to perform a discussed attack step without another tactic or technique, even if those dependencies were not explicitly written.

20 entries, which are then assigned to annotators for further analysis. We continued to follow the same procedures as before.

In rare cases, relevant labels were missing from the top-20 predictions, but the annotators were not explicitly instructed to manually include those labels in the dataset. Thus the recall of the annotations is inherently imperfect, and the labels tend to be biased towards to the use of InfoNCE, Nevertheless, based on the annotators' subjective assessment, the estimated annotation recall ranged from 95-100%, indicating that this dataset deviates minimally from a perfect annotation. Consequently, this split contains a significantly higher number of labels per sample compared to competing datasets., e.g., TRAM.

In conclusion, our Expert dataset, and particularly the test split, is of relatively small size, but is comprised of fully representative text paragraphs and has exemplary annotation precision and recall.

D Further Experimental Studies

D.1 Metrics

The definitions of the used metrics in our experiments are reported below.

P$@k$. Given a ranked list of predicted labels for each sample, the micro precision of the top-k is defined as: $P@k = 1/k \sum_{i=1}^{k} 1_{y_i^+}(l_i)$, whereas l_i is the i-th label in the ranking and $1_{y_i^+}$ is the indicator function.

R$@k$. Similarly, the micro recall of the top-k is defined as: $R@k = 1/|Q| \sum_{i=1}^{k} 1_{y_i^+}(l_i)$, whereas $|Q|$ is the number of positive labels in the sample.

F1$@k$. The metric maintains the harmony between P$@k$ and R$@k$ of a given ranked list, and is calculated as $\frac{2 \cdot P@k \cdot R@k}{(P@k+R@k)}$.

MRR$@k$. The metric measures the relative ordering of a ranked list, with RR is the inverse rank of the first relevant item in the top-k ranked list. Accordingly, MRR$@k$ is measured as follows. $MRR@k = 1/S \sum_{i=1}^{S} 1/rank_i$, whereas S is the number of samples.

D.2 Training Procedure and Hyperparameters

While InfoNCE and @-balanced are with normal training procedures, to leverage the effectiveness of the *asymmetric* loss, which performs optimally under stable gradient conditions, we adopt a two-step training procedure in our experiments. Initially, the model is trained using an @-balanced loss. Once the training process reaches a stable state, we then introduce the *asymmetric* loss.

We report the best hyperparameter sets for all models in Table 5. For the XMTC baselines, the parameter ranges for the probabilistic-based tree construction (i.e., with Huffman or K-Means) are designed to closely resemble the structure of the ATT&CK taxonomy. This resemblance is achievable thanks to its dot-separated naming convention, where the prefix represents the super technique.

D.3 Analysis

We report in Table 7 the models' results in the *technique*-level of the label hierarchy, where a subtechnique label is resolved to its technique. This is also a common practice in literature to streamline the complexity of the task. Overall, all models present significant improvements and our proposed ones are also benefited in this setting.

To further examine the difficulties posed by the Expert Dataset, we present the outcomes of models trained on the training splits of *Procedure*+ and TRAM, evaluated on the entire Expert dataset. The results are showcased in Tables 6 and 7. Overall, although all models exhibit reduced performance in this scenario, our models demonstrate superior generalization capability. Also, InfoNCE performs rather robustly in this setting, perhaps due to its stable nature to noisy input representation stemming from long-form text.

D.4 The Large Language Model (LLM) Dilemma.

In this section, our aim is to explore the extent to which information extraction tasks, particularly TTP mapping, can be addressed by LLMs, e.g., GPT-4 or PaLM-2.

In the pursuit of this target, we carefully crafted a setting to prompt ChatGPT (public version) for the a handful of samples in the *Expert* dataset. In general, the responses provided by Chat-GPT are remarkable and somewhat persuasive in certain instances. However, it is evident that the answers primarily consist of high-level information (sometimes hallucinatory), with a lack of granularity that makes it useful, e.g., for accurate modeling of the attack steps. Our objectives are, on the other hand, to precisely map to (sub-) techniques so that to reveal the actual capabilities of a threat group or a particular attack.

Our findings are:

Table 5: The default hyperparameters used in the experiments for each model.

	Models	**Hyperparams**
Ours	@-balanced	{cls-ratio: {γ: 0.11}}
	InfoNCE	{cls-ratio: {γ: 1.}}
	asymmetric	{γ_pos:1, γ_neg:3, cut-off: 0.1}
	- base settings	{learning_rate: 1e-3, auxiliary_task: {α: 0.6, β: 0.4}, batch_size:[2,**4**,8], negative_samples:[**30**,60] sampling_method: *random*}
	- auxiliary	{α: 0.6, β: 0.4}
	Dynamic Triplet Loss	{cls-ratio: {γ: 0.11} learning_rate: 1e-3, auxiliary_task: {α: 0.6, β: 0.4}, batch_size:[2,**4**,8], negative_samples:[**30**,60] sampling_method: *random*}
	NAPKINXC	{model: PLT, tree_type: {“hierarchicalKmeans”, “huffman”}, arity:{2,10, 20}, max_leaves:{10, 20}, kmeans_eps=0.0001, kmeans_balanced={True, False}}
	XR-LINEAR	{mode: “full-model”, ranker_level: 1, nr_splits: 16}
	XR-TRANSFORMER	{mode: “full-model”, negative_sampling: [“tfn”, “man”], , do_fine_tune: True, only_encoder: False}
	ExtremeText + Sigmoid	{loss: sigmoid, neg: [0, 100], tfidfWeights: True}
	ExtremeText + PLT	{loss: “plt”, neg: [0, 40], tree_type: {“hierarchicalKmeans”, “huffman”}, tfidfWeights: True}

Table 6: Results on the *entire* Expert dataset, trained on the training splits of *Procedure*+ and Tram. Bold denotes best-performer.

Methods	**P@1**	**R@1**	**F1@3**	**MMR@3**	**F1@5**	**MRR@5**
TTPDrill (BM25)	.311	.166	.226	.364	.207	.375
NAPKINXC	.43	.186	.3	.51	.275	.519
XR-LINEAR	.426	.198	.311	.517	.275	.529
InfoNCE	**.489**	.208	**.362**	**.564**	**.339**	**.576**
@-balanced	.443	.195	.328	.528	.324	.543
Asymmetric	.484	**.217**	.348	.558	.333	.573

Table 7: **Technique-level** results on the *entire* Expert dataset, trained on the training splits of *Procedure*+ and Tram. Bold denotes best-performer.

Methods	**P@1**	**R@1**	**F1@3**	**MMR@3**	**F1@5**	**MRR@5**
TTPDrill (BM25)	.369	.202	.283	.437	.267	.449
NAPKINXC	.51	.26	.344	.583	.375	.592
XR-LINEAR	.526	.279	.378	.595	.332	.609
InfoNCE	**.556**	.286	**.447**	**.621**	**.432**	**.633**
@-balanced	.506	.273	.428	.594	.429	.604
Asymmetric	.543	**.287**	.442	.615	.423	.626

- Chat-GPT: generate an answer that derives from the problem description, so-called prompt, which is rather non-*deterministic*, with no fixed answer for the same problem. This is partially due to the non-*constrained* answer search space.

- What we need: generate best-matched TTPs by comparing the given text to the ATT&CK TTP collection (bounded, deterministic). However, LLMs in general have the potential to alleviate this issue to some extent if they undergo effective domain-specific training during the pre-training phase and are subsequently fine-tuned using high-quality datasets, such as an enhanced version of our Expert, in the *reinforcement-learning-from-human-feedback* phase (Karpathy, 2023).

- Currently, Chat-GPT (3.5) is not up to the task.

Here are a couple of examples:

> **Q1**: What MITRE ATT&CK techniques (TTPs) are mentioned in the following text: “*After the .NET PE file has been run, we observed the same behavior as the above QUADAGENT sample of dropping a PowerShell script with the filename SystemDiskClean.ps1 alongside a VBScript file with the same name. The C2 techniques remained identical, with the only change being the server which became cpuproc[.]com.*”

Table 8: **Technique-level** (wherein a sub-technique is resolved to its super-technique, a common practice in literature to lower the task's complexity) results of all models on 3 datasets. Bold denotes *best* while underscore signifies *second-best* performance. Indented (−) denotes training without the specific option wrt. the preceding model. *Ideal* R@1 on the Expert dataset is 0.504. $^{\mathcal{T}}$ uses pre-trained SecBERT.

		Procedures+				TRAM				Expert			
	Methods	P@1	R@1	F1@3	MRR@3	P@1	R@1	F1@3	MRR@3	P@1	R@1	F1@3	MRR@3
Baseline	TTPDrill (BM25)	.294	.290	.152	.297	.281	.271	.161	.295	.197	.096	.096	.279
	BR$^{\mathcal{T}}$	.409	.655	.285	.655	.399	.647	.279	.647	.167	**.295**	.117	.295
	Dynamic Triplet-loss$^{\mathcal{T}}$	.449	.447	.408	.539	.404	.353	.382	.513	.559	.166	.344	.631
XMTC	eXtremeText (Sigmoid)	.659	.649	.426	.713	.742	.704	.494	.793	.439	.212	.333	.521
	eXtremeText (PLT)	.644	.636	.403	.689	.714	.679	.464	.756	.465	.206	.327	.532
	NAPKINXC	.698	.687	.426	.764	.800	.748	.495	.864	.548	.253	.409	.626
	XR-LINEAR	.705	.700	.429	.772	.817	.765	.494	.870	.586	.261	.439	.669
	XR-TRANSFORMER$^{\mathcal{T}}$	.683	.673	.416	.747	.801	.750	.488	.856	.554	.245	.405	.633
Ours	InfoNCE$^{\mathcal{T}}$	.759	.727	.624	.823	.819	.696	.668	.876	.741	.228	**.515**	**.871**
	@−balanced$^{\mathcal{T}}$	**.843**	.806	.666	.892	.889	.778	.711	.927	.731	.224	.491	.789
	(−) auxiliary	.714	.689	.579	.791	.817	.697	.648	.88	**.754**	.233	.509	.816
	(−) Transformers	.777	.733	.664	.86	.791	.683	.713	.875	.718	.226	.497	.782
	Asymmetric$^{\mathcal{T}}$	.841	**.806**	**.677**	**.892**	**.903**	**.789**	**.726**	**.938**	.745	.236	.483	.802

A1: The text mentions the following MITRE ATT&CK techniques:

- File and Directory Discovery: The QUADAGENT sample drops a PowerShell script and a VBScript file with the filename SystemDiskClean.ps1.
- Command and Control (C2) Communications: The text indicates that there is a change in the C2 server from appexsrv[.]net to cpuproc[.]com.

It's important to note that there may be additional MITRE ATT&CK techniques employed in this attack, but they are not mentioned in this particular text. A more detailed analysis would be required to confirm the full set of techniques used in this attack.

Implications: The answer from ChatGPT is rather generic, while missing out the compelling Masquerading technique.

Q2: What MITRE ATT&CK techniques (TTPs) are mentioned in the following text: "Russian state-sponsored APT actors have exfiltrated credentials and exported copies of the Active Directory database ntds.dit."

A2: The following MITRE ATT&CK TTPs are mentioned in the given text:

- Exfiltration: The adversaries have exfiltrated credentials and data from the victim's network.
- Collection: The adversaries have exported copies of the Active Directory database ntds.dit, which is a technique used for data collection.

Implications: Chat-GPT gives a rather generic answer, and cannot point to specific TTPs (i.e., NTDS or OS Credential Dumping).