# CyBERT: Contextualized Embeddings for the Cybersecurity Domain

Priyanka Ranade, Aritran Piplai, Anupam Joshi, Tim Finin
Dept. of Computer Science & Electrical Engineering, University of Maryland, Baltimore County,
Email: {priyankaranade, apiplai1, joshi, finin}@umbc.edu

***Abstract*—We present CyBERT, a domain-specific Bidirectional Encoder Representations from Transformers (BERT) model, fine-tuned with a large corpus of textual cybersecurity data. State-of-the-art natural language models that can process dense, fine-grained textual threat, attack, and vulnerability information can provide numerous benefits to the cybersecurity community. The primary contribution of this paper is providing the security community with an initial fine-tuned BERT model that can perform a variety of cybersecurity-specific downstream tasks with high accuracy and efficient use of resources. We create a cybersecurity corpus from open-source unstructured and semi-unstructured Cyber Threat Intelligence (CTI) data and use it to fine-tune a base BERT model with Masked Language Modeling (MLM) to recognize specialized cybersecurity entities. We evaluate the model using various downstream tasks that can benefit modern Security Operations Centers (SOCs). The fine-tuned CyBERT model outperforms the base BERT model in the domain-specific MLM evaluation. We also provide use-cases of CyBERT application in cybersecurity based downstream tasks.**

## I. Introduction

Transformer-based natural language models such as Bidirectional Encoder Representations from Transformers (BERT) [1] and GPT [2] are state of the art methods that enable computers to process text data. While the decoder-based GPT is constrained to generative natural language use cases, BERT is an encoder-based transformer model trained on the entire English Wikipedia [3] and Brown Corpus [4]. BERT provides substantial results across tasks such as machine translation [5], natural language inference [6], and question answering systems [7], due to its bidirectional and context-dependent architecture. In addition, there exist a variety of large, general-domain *pre-trained* BERT models re-purposed for a multitude of downstream tasks [8].

Natural language tasks that require processing over a large corpus of general text data benefit tremendously from the pre-trained BERT models. In 2019, Google introduced BERT into its search engine, and has expanded its usage and testing over the past two years [9]. Through query sampling and rank-based evaluation, they determined that BERT provides proven improvements in tasks such as search, document retrieval, and ranking across a diversity of general-domain text. Google compared the relevance of the returned query responses using the BERT-based search engine and the traditional search engine which primarily uses PageRank. The main improvement comes from BERT's bidirectional architecture and enhanced attention mechanism, giving it the ability to handle *conversational queries*. Conversational queries are those that are entered into the search engine as full *sentences* rather than as a a collection of related *keywords*. Using BERT to process conversational queries enables a search engine to understand a user's query more like a person would and produce better retrieval results.

Though there are clear benefits of using the general pre-trained BERT models for domain-independent use cases, they do not provide state of the art results for tasks that require domain-specific, fine-grained text. The word distributions of the general corpus on which BERT was trained on (Wikipedia and the Brown Corpus) differ tremendously from many domain-specific corpora, such as those in cybersecurity and medicine. Directly applying general language models like BERT to domain-specific text mining tasks has many limitations, such as the inherent inability to represent domain-specific terms that it is unfamiliar with (terms and context that do not exist in the general corpus used to train the model).

To address this problem, many researchers use an approach called *fine-tuning* to seed general pre-trained transformer models like BERT with additional text from the domain to better adapt the model to a domain-specific tasks [10]–[12]. In this way, the general transformer model can learn domain specific features through minimal weight adjustments, without undergoing the resource intensive task of training from scratch.

For example, BioBERT [13] is a domain-specific BERT model, fine-tuned on large-scale biomedical corpora. BioBERT significantly outperformed the general BERT model on biomedical named entity recognition, relation extraction, and question-answering tasks.

In the similar way that bio-medical tasks rely on adapted word representations, cybersecurity domain texts also contain a drastically different vocabulary than that of the corpus used to train the original BERT model. As the amount of Cyber Threat Intelligence (CTI) continues to increase on the web, cybersecurity text mining is becoming an increasingly integral task in Security Operations Centers (SOCs). Cybersecurity professionals utilize information gained from CTI sources to aid in vulnerability management, threat filtering, and adversarial behavior prediction. There are many benefits to the increasing presence of CTI sources across the web, such as access to large-scale unlabeled training data. However, the detriment to the large amount of CTI available is the inability for cybersecurity professionals to gain valuable and actionable insights across all sources. There have been several models and pipelines proposed to ease the information overload of cybersecurity professionals, and shift the processing load to natural language models [14]–[19]. CyBERT can support existing pipelines by providing the ability to contextually understand cybersecurity threats, exploits, attack-vectors, and adversaries.

CyBERT can do this by contextually learning fine-grained cybersecurity domain information. For example, as human cybersecurity professionals read CTI samples, a diversity of knowledge is required to gain actionable insight on the information. For example, when reading about a malware related exploit, a series of deductions are made to understand the type of malware, its probable destinations, and behavior. Cybersecurity threat hunters will typically use context clues in the CTI to examine the exploit. Similarly, CyBERT will be able to use both general knowledge as well as linguistic rules to reason about the exploit and its likely relations.

There is no public model to our knowledge that uses a BERT-based approach to represent cybersecurity text data. The primary contribution of this paper is the development of CyBERT, which is a BERT model fine-tuned with Masked Language Modeling (MLM) as well as for several cybersecurity-related downstream tasks. CyBERT can be beneficial for the cybersecurity community due to its ability to adapt to a diversity of tasks that several groups of organizations, cybersecurity professionals, and tools can integrate. We provide examples of use-cases by testing CyBERT on cybersecurity NER, Multi-class classification, and Cybersecurity Knowledge Graph (CKG) improvement downstream tasks.

Our work makes three main contributions:

- CyBERT: A Pre-trained contextualized embedding for the cybersecurity domain (Section III-C2),
- Labeled Datasets to fine-tune related cybersecurity downstream tasks (Section III-A, III-C1 and IV), and
- Analysis of CyBERT performance on NER, Multiclass Classification, and Cybersecurity Knowledge Graph completion (Section IV).

The organization of this paper is as follows. In Section II we provide an overview of the related work and background concepts on textual cybersecurity data, natural language models, and the fine-tuning process. Section III includes the methodology for creating a cybersecurity corpus to fine-tune BERT, developing a cybersecurity MLM dataset, and fine-tuning BERT with MLM. We showcase the fine-tuned CyBERT on a variety of cybersecurity specific down-stream tasks in Section IV and conclude and discuss future work in Section V.

## II. Related Work

### A. Leveraging Textual Cybersecurity Data

Developments in cybersecurity attacks, vulnerabilities, and threat actors has led to a shift in developing further, more sophisticated methods for safeguarding critical systems. The integration of machine learning and artificial intelligence for cybersecurity tasks has grown over the past decade. Examples of AI-enabled cybersecurity tasks include enhanced intrusion detection and prevention systems [20], attack prediction strategies [21] and malware analysis [22]. Examples of type of training data used for these systems include server and application logs, malware binaries, and network traffic. A large portion of this data is open sourced, and is often referred to as Cyber Threat Intelligence (CTI). This data can be collaboratively shared with the greater public for precaution, use in research, or as part of an integrated patch management system found in modern Security Operations Centers (SOCs). In addition, a large amount of CTI is also found in the form of unstructured, and semi-structured *text*. Textual cybersecurity information is most often found on security blogs, social media, conversation forums, the dark web, as well as within several github repositories [23]. This information is typically documented by subject matter experts such as cyber analysts, security engineers, and researchers, within academia and the greater public and private technology sectors.

Though the security community benefits tremendously from open sourced CTI, this information, whether it is text or other forms of cybersecurity data, is typically spread across multiple sources, providing difficulty in providing quick *and* insightful information to the automated systems that require the information for operation. Several AI-based cybersecurity systems have been developed to integrate disparate cybersecurity information. For example, Mittal et al. developed *CyberTwitter*, which mines threat intelligence from Twitter and generates actionable alerts for cybersecurity threats and exploits [19]. Similar methods have been developed that leverage a Cybersecurity Knowledge Graph (CKG) to represent and store cybersecurity intelligence to be used as training data for AI-based cybersecurity systems, or from extraction or ingestion pipelines [14]–[18], [24], [25].

One of our goals for creating CyBERT is to provide the security community with state-of-the art, continuously updated cybersecurity langauge models that can be used for a variety of cybersecurity specific downstream tasks. Systems such as those described above, can leverage the CyBERT model for capabilities such as Named Entity Recognition (NER), intelligence classification, and knowledge augmentation and integration.

The next section describes the evolution of natural language models, as well as the state-of-the art architecture of BERT.

### B. Evolution in Natural Language Processing Models

There are a variety of Natural Language Processing (NLP) techniques for representing textual data. The most basic models include bag of words (one-hot encoding) where each element in a vector maps to a unique word or token (n-gram) in the vocabulary of a corpus [26]. Though these techniques are still powerful for tasks such as search, they are often used in conjunction with modern models that have the ability to capture the semantics of the words, as well as with more computational efficiency. Distributional similarity based representations such as Word2Vec [27] and GloVE [28] are examples of such models and are often referred to as *neural word embeddings*. Neural word embeddings are dense representations where each word is represented by a real-valued vector. Words are also represented using *context similarity*. This means that words that have similar semantic meaning, will have a similar representation.

These methods remained the state of art until language models such as transformers like BERT [1] and GPT [2] were developed. Unlike neural word embeddings, language modeling trains with deep, hierarchal representations that can capture contextual, high-level information such as long-term dependencies and *sequential context*. Models that are pretrained with language modeling can later be fine-tuned for domain-specific downstream tasks. In our previous work, we fine-tuned the GPT-2 transformer model with cybersecurity text and generated fake textual cybersecurity CTI as part of a data-poisoning experiment [12].

### C. BERT and Transfer Learning

BERT is a type of encoder transformer language model, which maps an input sequence, such as a sequence of words (sentence), into a high dimensional space (n dimensional vector) [1]. Other transformer models such as GPT, are examples of decoder models which take the vector from the encoder and produces an output sequence (prediction) for a given task. Transformers are therefore upgraded versions of traditional encoder-decoder models that do not use Recurrent Neural Networks (RNN), but rely solely on a multi-head *attention* mechansim to capture significant semantic features in a given input sequence. Applications of BERT specifically, include text classification, summarization, paragraph-based question-answering, and information retrieval.

In this paper, our goal is to fine-tune state of the art pre-trained BERT model with cybersecurity domain information and leverage the model for cybersecurity tasks. Fine-tuning is an example of transfer learning, a method that applies existing knowledge to a new, but similar problem. The pre-trained BERT model is popularly used as a base model that provides general features, but can be augmented for specific tasks [10]–[12].

## III. Methodology

We create cybersecurity bidirectional encodings by fine-tuning the base BERT model (*bert-base-cased*) with Masked Language Modeling (MLM), utilizing a large corpus of unlabeled cybersecurity text. The fine-tuning approach allows us to yield higher accuracy with much less data and training time in comparison to learning from scratch. The resulting fine-tuned model can be used for tasks such as vulnerability and exploit search, cybersecurity based-NER, and cybersecurity knowledge graph (CKG) completion [18], [29].

The methods for our approach are displayed in Figure 1. We first create a cybersecurity corpus by collecting unlabeled, textual cybersecurity data from a variety of sources such as open source blogs [30] standardized by the security community, National Vulnerability Database (NVD) [31], and Common Vulnerabilities and Exposures (CVE) [32] records. Using the cybersecurity corpus, we *extend* the base BERT model's vocabulary to include fine-grained cybersecurity domain entities. Once the vocabulary is extended to fit the cybersecurity domain, and the embedding space updated, we use the cybersecurity corpus to *fine-tune* the *pre-trained* base BERT model (bert-base-cased) with Masked Language Modeling (MLM).

CyBERT leverages the basic architecture of the base BERT model, *bert-base-cased*. Details on the bert-base-cased architecture can be found in Section III-C2. We discuss more thoroughly the creation of the cybersecurity dataset used to fine-tune the base BERT model in Section III-A, the process of using fine-tuning with MLM in section III-C1 and III-C2 and later discuss methods for fine-tuning CyBERT for various downstream text mining tasks in Section IV.

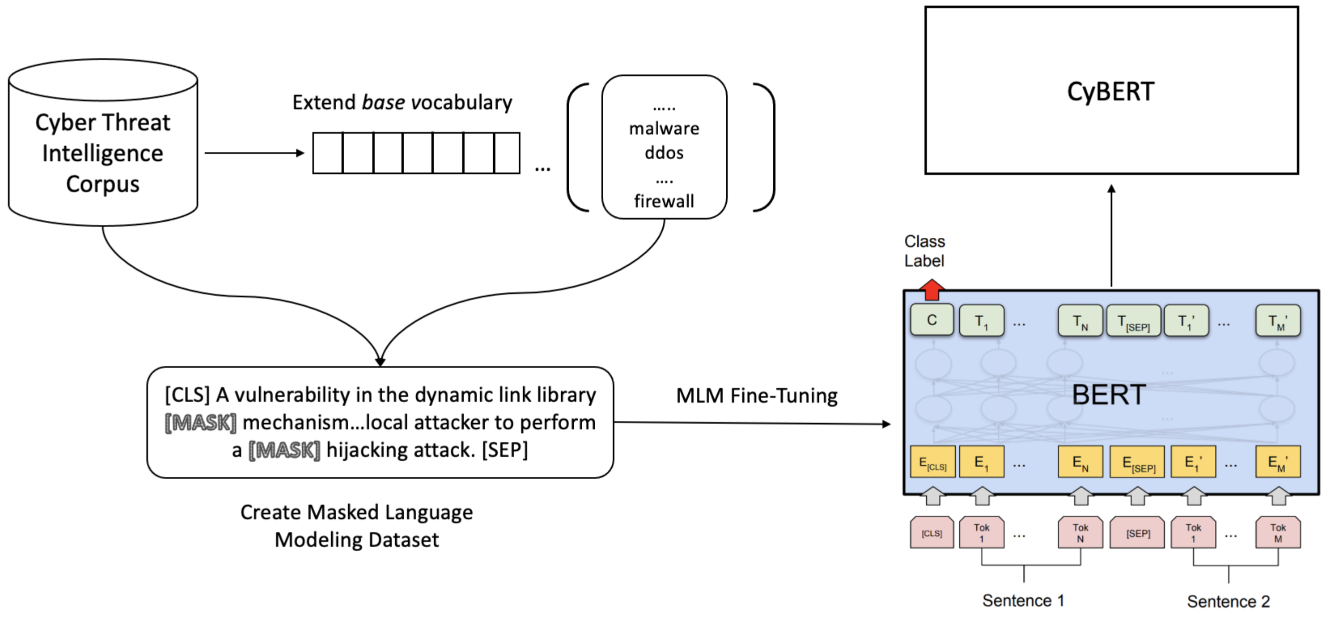
### A. Cybersecurity Corpus Creation

Fine-tuning a base BERT model requires an extensive domain dataset. In order to train BERT to better recognize cybersecurity domain information, we created a cybersecurity text corpus to fine-tune the base BERT model. Our corpus includes a combination of free text and semi-structured text data, which contains both general and localized, fine-grained cybersecurity domain information.

Our corpus can be considered the gold standard for cybersecurity data due to the inclusion of widely utilized and referenced resources used in the cybersecurity community. The free-text examples in our repository are composed of news articles and technical reports found on open sources such as social media sites like Twitter[1], and repositories such as vendor-managed documentation hubs. Security news articles are reliable resources used by cybersecurity analysts and engineers to stay informed on emerging vulnerabilities and exploits. A considerable fraction of the security news article data in our cybersecurity corpus comes from *Krebs on Security*, which is a global resource referenced by Security Operations Centers (SOCS) at a variety of organizations [30]. It is also referred to by multiple security bloggers who use the resources to provide current, up to date information to their followers. Krebs on Security is composed of reports describing recent, medium to high impact exploits and vulnerabilities detected by security analysts either in the wild, or within organizations and includes both general and fine-grained cybersecurity exploit and vulnerability information.

On the other hand, the resource *APT Reports* is a technical repository that focuses on documenting low-level information on malware-specific attacks and Advanced Persistent Threat (APT) groups [33]. An example of a more structured form of cybersecurity data includes the *CVE database*, maintained by MITRE Corporation [32]. This data is structured in both JSON and XML formats and contains structured fields to segment the data easily. Some examples of the fields are ”attack means”, ”product name”, ”exploit target”, and ”vulnerability description”. *CVE database* is a common resources used by corporations to track vulnerabilities and exploits associated

[1] https://www.twitter.com



[CLS] A vulnerability in the dynamic link library [MASK] mechanism...local attacker to perform a [MASK] hijacking attack. [SEP]

Fig. 1. Architecture Diagram for fine-tuning of the pre-trained BERT model to create CyBERT.

with popular products they produce and utilize. CVE data is also commonly ingested in SOC Intrusion Detection and Prevention Systems (IDS and IPS) [34]. The CVE datasets have also aided us in creating Named Entity Recognition (NER) datasets, described in Section IV.

By including both general and fine-grained CTI, we can fine-tune the general BERT model to learn a diversity of domain cybersecurity information that is otherwise not present in the original corpus utilized to train BERT.

The security news category contains 500 articles from Krebs on Security. The vulnerability reports contain 16,000 Common Vulnerability and Exposures (CVE) records provided by MITRE Corporation and National Vulnerability Database (NVD) from years 2019-2020. Lastly, we collect 500 technical reports on APTs from the available *APTNotes* repository.

### *B. Vocabulary Extension*

Before fine-tuning BERT with Masked Langauge Modeling (MLM), we first extend the base model's vocabulary with domain specific words and resize the original embedding matrix to include unique cybersecurity tokens. During fine-tuning tasks, modifications are only made to the embedding space, but not to the tokenizer vocabulary. It is important for cybersecurity domain-specific tokens to be present in the vocabulary due to the *lack of* cybersecurity words and subwords in the base model tokenizer vocabulary. The base model tokenizer contains 28,996 words. During the MLM task, the model will randomly mask 15% of the tokens from an input sentence with the objective of predicting the original vocabulary ID of the masked word, based on term context. By adding this step before the MLM fine-tuning process, we can ensure that the MLM output will locate the the nearest predictions in BERTs vocabulary to the transformed vectors. The final hidden vectors are fed into a output softmax over the vocabulary. Therefore, the MLM predictions take into account the position of an input token in the sentence, and the location can be derived from BERT's vocabulary. BERT's extended cybersecurity vocabulary will represent top $k$ predictions for a word and subword at a particular location in an input sentence. The input term will be tokenized into words and subwords within BERT's extended tokenizer.

A resource intensive way of learning the vocabulary of a specific domain is through *pre-training* the model from scratch on a new corpus of domain-specific task, typically as large as the original Wikipedia and Brown corpus used to train the base BERT model [35]. An alternative is to use an *extension* method, where a domain-specific vocabulary can be added to an existing vocabulary. These terms will be randomly initialized when added, but will be learned during the MLM fine-tuning process [36].

We use the extension method in order to add cybersecurity domain tokens to the original tokenizer. The first step to extend the vocabulary is to develop a list of ranked tokens from the cybersecurity corpus. We first utilize a word tokenizer from spaCy [37] to extract tokens within the cybersecurity corpus. We then find the inverse document frequency (IDF) for each token, and rank the tokens in order of most to least significant. We add the 1000 highest IDF rank words to the base vocabulary. The size of CyBERT vocabulary is therefore 29,996 terms.

We then resize the embedding matrix of the BERT model to match the extended vocabulary size. During the MLM tuning process, new embedding vectors will be added to represent the extended vocabulary with cybersecurity terms. Once the vocabulary is updated with common cybersecurity terms and the embedding matrix of the model is resized, we can start fine-tuning the model on the cybersecurity corpus using MLM. The MLM fine-tuning process for CyBERT is described in the next section.

### *C. Fine-Tuning with Masked Language Modeling*

The BERT architecture is a transformer encoder model that is able to contextually process text data. BERT expands previous transformer architectures by introducing bi-directional training, allowing words to be defined by their context rather than pre-fixed definition. Specifically, BERT uses a method called *Masked Language Modeling* (MLM) to learn an internal representation of an input, such as a sentence. With this method, BERT does not process the input from left to right to predict the next word, but rather, an entire sentence is fed as input to BERT, where some tokens in the model will be randomly (or selectively) removed. The task for BERT is to identify an appropriate word to *fill the blank* in a given input sentence.

The CyBERT model re-applies MLM on the general BERT model by fine-tuning with the cybersecurity corpus described in Section III-A. We can then use CyBERT for many related downstream tasks such as cybersecurity Named Entity Recognition (NER), threat classification, and Cybersecurity Knowledge Graph (CKG) completion, which is described in Section IV.

The process of fine-tuning with MLM is displayed in Figure 1. We first split the text corpus into train and test MLM datasets, where the test dataset contains *randomly masked* tokens and the train dataset contains unmasked counterparts for the respective test samples. We then use the datasets to train the base BERT model to learn an internal representation of the cybersecurity corpus during the fine-tuning process. These steps are more thoroughly described in the following sections.

*1) Creating a Cybersecurity Masked Langauge Modeling Dataset:* We first use the cybersecurity corpus described in Section III-A to create a training dataset to fine-tune the general BERT model using MLM. For each text file in the corpus, we separate the text into a set of sentences and employ standard text cleaning techniques (removing special characters, converting to lowercase, and removing sentences less than 50 characters). We were able to retain more than 70% of the cleaned corpus. The list of cleaned, split lines, will serve as the *labeled* dataset for the MLM task described in Section III-C2. The test set for the MLM training contains samples of masked sentences for each CTI sentence in the labeled set. More specifically, for each sentence in the labeled set, we create a corresponding masked sample by removing 15% of the tokens in the sentence, at random. An example of a masked data sample can be found in Figure 1.

*2) Training CyBERT:* We fine-tune the base BERT model with Masked Language Modeling (MLM) using the datasets created in the previous section (Section III-C1) to produce CyBERT. Using the *HuggingFace* library [38], we load the *bert-base-cased* pre-trained model as the base model. The *bert-base-cased* model has 12 layers (transformer blocks), a hidden size of 768, and 12 self-attention heads, and is case-sensitive. To avoid over-fitting we limit the MLM fine-tuning process to one epoch and set the sequence length to 128.

We first preprocess the cybersecurity corpus by using the *BertTokenizer* to split the CTI samples into a list of sentences. After tokenization, the text is divided into three tensors, *input ids, token type ids, and attention mask*. A copy of the input ids are added to a *labels tensor*, which is later used to calculate loss. We employ a random masking approach, and assign a 15% probability that a token will be masked (any token, except for the CLS and SEP tokens).

The input tokens produce a *logits* output embedding where the vector length is equal to the size of the extended vocabulary. The predicted token is extracted from logits using softmax and argmax transformation. The loss is calculated as the difference between the output probability distributions for each predicted token and the one-hot encoded labels. We achieve a loss value of 0.1451.

To further test the performance of CyBERT, we run a *fill-in-the-blank* assessment test the model's ability to fill in relevant terms on unseen CTI samples. We use the *fill function* from the HuggingFace library for the assessment. We also compare the output tokens to the base BERT model. We randomly mask attack vector, vulnerability, and adversary terms in the experimentation set consisting of fifty CTI samples. The fill function returns the top 5 predicted tokens for each masked sample.

CyBERT consistently filled in blanks with either correct term (as the highest rank output token) or with a term that provides a similar contextual meaning. In addition to the highest rank output token, CyBERT also consistently returned contextually relevant terms in its top 5, in comparison to BERT, which at times returned non-cybersecurity related (general) terms.

For example, for the following CTI sample,

> *'Easily exploitable vulnerability allows low privileged attacker with [MASK] access via multiple protocols to compromise MySQL Server.'*

CyBERT and BERT predict the top five fill-in tokens for the *[MASK]* placeholder. A comparison of CyBERT and BERT output tokens for the fill-in-the-blank assessment in Table I.

CyBERT was able to predict the correct term with 98% confidence. In addition to predicting the correct term, CyBERT returned contextually relevant terms that pertain to the context of the exploit. The MySQL server attack is an example of privelege escalation, relevant to the third highest ranked word (root). In addition, this was a network-based attack, so the second highest term (local) and fourth highest term (remote) are relevant. This example shows that CyBERT was able to determine the MySQL server attack was a remote, privilege escalation attack and fill in the correct terms. BERT did not have the same internal representation of domain-specific cybersecurity information and as a result, did not return any

| CyBERT | BERT (*bert-base-cased*) | CyBERT Score | BERT Score |
|---|---|---|---|
| **network** | easy | 0.9825 | 0.1802 |
| local | multiple | 0.0007 | 0.0709 |
| root | direct | 0.0008 | 0.0327 |
| remote | secure | 0.0007 | 0.0239 |
| system | unlimited | 0.0007 | 0.0231 |

TABLE II
CYBERT AND BERT RANKED PREDICTIONS FOR CTI FILL-IN-THE-BLANK TASK. 'NETWORK' IS THE CORRECT RESULT IDENTIFIED BY CYBERT.

fine-grained cybersecurity terms, but rather generic terms that do not properly indicate a remote privilege escalation attack.

## IV. Downstream Tasks with CyBERT

The cybersecurity community can leverage several downstream NLP tasks to augment security workloads in traditional Security Operations Centers (SOCs). In this section, we describe several methods in which *CyBERT* can aid in various SOC-related activities such as cybersecurity based Named Entity Recognition (NER), threat classification, and Cybersecurity Knowledge Graph (CKG) completion. Each of the above mentioned tasks are more discretely described in the following sections. For each of the tasks, we further fine-tune CyBERT to fit the particular use-cases listed above.

### A. *Cybersecurity Named Entity Recognition*

Named Entity Recognition (NER) can be beneficial for the cybersecurity community in a variety of ways. In particular, NER models allow cybersecurity professionals to efficiently extract relevant attack and vulnerability information from unstructured textual Cyber Threat Intelligence (CTI). CTI text contains specialized domain entities such as *exploit target* and *attack means* which are not interpreted by models trained with general-purpose NER datasets.

Improving NER tasks for the cybersecurity domain can increase situational awareness within an organization by automating information retrieval and extraction tasks across large quantities of CTI. This allows security analysts to efficiently gain a high level view and understanding of general exploit patterns, attack schemas, and potential vulnerabilities. In a SOC setting, these NER models can integrate cybersecurity-based search with general search engines as well as domain-specific cybersecurity engines such as Shodan [39], to improve rankings and relevance of exploit, attack vector, and vulnerability search results by grouping CTI samples based on similarity.

There is a lack of large-scale public cybersecurity text analytics datasets available in the public. Harvard Dataverse published a cybersecurity-based NER dataset, containing domain labels for 1000 cybersecurity tweets [40]. Similarly, we have also developed a cybersecurity NER dataset in previous research [41]. Though both of the datasets contain labeled entities specialized to the cybersecurity domain, they differ in their specificity. While the Harvard Dataverse dataset contains more broad cybersecurity labels (aimed for social media studies) our dataset contains fine-grained labels that would be found on sources such as CVE and NVD. For this study, our goal is to train BERT to recognize a variety of fine-grained cybersecurity entity types. Our cybersecurity corpus also primarily contains fine-grained information. Therefore, we fine-tune CyBERT using our previously developed NER model and evaluate CyBERT against low-level cybersecurity entities.

| Named Entities |
|---|
| Attack-Pattern |
| Course-of-Action |
| Exploit-Target |
| Malware |
| Software |
| Version |
| Vulnerability |

TABLE III
LABELED ENTITIES IN CYBERSECURITY NER DATASET.

The Cybersecurity NER dataset was created through supervised methods using the BRAT rapid annotation tool [42]. We provided a group of three human annotators approximately 500 samples of unstructured text files from the following sources:

- Common Vulnerabilities and Exposures [32]
- Microsoft Bulletins [43]
- Adobe Bulletins [44]
- After Action Reports [45]

The annotators populated their annotations using the community standard, BRAT rapid annotation tool, which is an online environment used for collaborative text annotation [42].

For each unlabeled sample, we assigned three annotators. We used the *inter-annotator agreement* of 66% (2/3 annotators must agree) to determine the most accurate labels. The final NER dataset includes 46,838 labeled entities and corresponding terms. The labeled entities in our NER dataset are displayed in Table IV-A.

We used the NER dataset to fine-tune CyBERT for a cybersecurity NER task. We first split the 46,838 labels into training (train), development (dev), and testing (test) datasets, using a 70/15/15 split respectively. Once we split the text, we extract the NER entity classes across the three files and dump them into a *labels* file, later used for NER fine-tuning. We set a maximum sequence length to 128, batch size to 16, a learning rate to 0.01. We train the model for four epochs.

The F1, precision, recall, and loss metrics for the test and dev sets are displayed in Table VI. After fine-tuning CyBERT on a cybersecurity NER task, we can use it to recognize named entities in unseen cybersecurity texts. For example, for the following CTI sample,

> *'CVE-2014-0137 SQL Injection Vulnerability in the saved_report_delete action in the ReportController in Red Hat CloudForms Management Engine (CFME) before 5.2.3.2 allows remote authenticated users to execute arbitrary SQL commands via unspecified vectors related to MiqReportResult.exits.'*

CyBERT recognized the following entities, shown in Table IV. In addition to sampling CyBERT, we also provided the same sample to an annotator. The labeled entities matched CyBERT's labeled entities, except for discrepancies between software and attack-pattern. For example, the annotator labeled

| Word | Predicted Entity | Score |
|---|---|---|
| CVE-2014-0137 | Vulnerability | 0.9916 |
| SQL | Software | 0.9933 |
| Injection | Attack-Pattern | 0.9952 |
| saved_report_deleteaction | Exploit-Target | 0.9958 |
| ReportController | Software | 0.9958 |
| CFME | Software | 0.9947 |
| 5.2.3.2 | Version | 0.9932 |
| commands | Attack-Pattern | 0.9763 |
| MiqReportResult.exits | Exploit-Target | 0.9922 |

TABLE IV
NER PERFORMANCE FOR UNSEEN CVE SAMPLE.

| Data Set | F1 | Precision | Recall | Loss |
|---|---|---|---|---|
| **Test** | 0.811 | 0.802 | 0.823 | 0.531 |
| **Evaluation** | 0.879 | 0.874 | 0.883 | 0.414 |

TABLE VI
TEST AND EVALUATION METRICS FOR CYBERT NER.

SQL and Injection as *Attack-Pattern*, while CyBERT labeled SQL as *Software*, with with high confidence. On the other hand, CyBERT labeled *Injection* correctly, and in a real world SOC context, SQL and Injection would appear as a single entity. Therefore, this particular discrepancy would not negatively impact SOC operations in normal operation. However, we are actively working on methods to reduce such instances and improve CyBERT entity prediction by creating additional NER datasets to provide CyBERT with diverse sets of examples and more training data.

### B. Multi-Class Classification

Another task that is particularly helpful for the cybersecurity community is multi-class classification. We further fine-tune CyBERT to perform multi-class classification. There are several use-cases where multi-class classification can be beneficial for cybersecurity operations. Cybersecurity analysts and threat hunters can use automated classification tooling to categorize types of attacks, vulnerabilities, adversaries, as well as attack-vectors. In particular, tools that can predict potential attack vectors given *textual descriptions* of vulnerabilities can automatically create mappings of attack and vulnerability pairings, which can later also be used as training data for AI-based cyber defense systems such as a Cybersecurity Knowledge Graph (CKG). This can reduce workloads for analysts who typically need to sift through attack details to categorize various types of attack vectors.

To model this scenario, we created an *Attack Prediction* dataset using 5,000 CVE records that can be used to fine-tune CyBERT for multi-class classification of exploit types and vulnerabilities. The dataset contains two columns, Description (DESC) and Exploit (EXPLT). The description column contains vulnerability details, and the exploit column contains a category in which the vulnerability description falls under. We named the categories using CVE's vulnerability and exploit classes. We tested with the following five category types:

- DoS
- SqlInjection
- XSSDirectoryTraversal
- BypassCredentials
- InformationGain

| DESC | EXPLT |
|---|---|
| The Pallets Project Flask version Before 0.12.3 contains a CWE-20:Improper Input Validation Vulnerability in Flask that can result in large amount of memory usage. | DoS |

TABLE VII
ATTACK PREDICTION DATASET SAMPLE.

A DoS CVE sample from the Attack Prediction dataset is shown in Table VII. We split the 5,000 records into a train/test set using a 70/30 split and radomly mix the dataset to ensure there are examples containing all five categories across both the train and test sets. Next, we format the dataset further to the features the out of box BERT classifier uses. The BERT classifier uses four features to make predictions: *guid*, which is an identifier that represents a unique instance in the dataset, *text_a* which is the text to be classified, *text_b* which denotes the relationship between sentences, and *label* which represents the classes a given text sample belongs to. We format the dataset using the guid, text_a and label fields. The guid is a numerical value that represents the sample number, text_a represents the attack description, and label represents the attack types. We use the BERT's tokenization package to perform preprocessing operations. The preprocessing steps include, converting whitespace characters to spaces, tokenizing the text, adding [CLS] and [SEP] tokens to mark the beginning and end of input sentences as well as mapping the words in the dataset to indexes found within the extended vocabulary we created in Section III-B. We set the sequence length to 128 and use the tokenizer to pad sentences larger than length 128. We train the model for three epochs and obtain a loss value of 0.0148 and a 98% evaluation accuracy.

Similar to the NER downstream task, the output of this system can be further be used as training input for various AI-based cyber defense systems. We discuss these examples in further detail in Section V.

## V. CONCLUSION

CyBERT is a fine-tuned BERT model trained on a large corpus of Cyber Threat Intelligence that contextually understands information in text about the cybersecurity domain. We fine-tuned the initial BERT model using Masked Language Modeling and an extended cybersecurity vocabulary in order to train it to learn an *internal representation* of fine-grained cybersecurity terms, concepts, and their relationships.

CyBERT can provide many benefits to the cybersecurity community, especially in Security Operation Center (SOC) settings that perform downstream tasks such as cybersecurity information extraction, attack prediction, and classification. We provide examples of CyBERT's performance on these tasks and its applications to real-world SOC settings by evaluating CyBERT on NER and attack classification tasks.

Once the cybersecurity entities have been identified using the CyBERT named entity recogntion, we can use them to create semantically rich Cybersecurity Knowledge Graphs (CKG). CKG allows effective use of CTI to evaluate and mitigate evolving risks. This helps solve organizational needs that require processing the raw data feeds and appropriately representing them. Knowledge graphs have become an important way to represent CTI due to their reasoning capabilities that are applied on the interconnected entities. In addition, the attack category output from the CyBERT classifier can be used for CKG entity clustering.

Another interesting downstream task that we are actively pursuing is using CyBERT to support Cybersecurity Knowledge Graph (CKG) completion. In order to first create and then complete these graphs, it is important to identify relations between different entities identified by the cybersecurity NER. Pingle et al. [17] provide a more traditional approach to an ontology based relationship extraction. In the same setup, CyBERT's contextualized embeddings can be used in place of the Word2Vec based cybersecurity embeddings. A simple feed forward network can then be trained to predict relations between entities. The resulting relations along with extracted entities can then be asserted into the cybersecurity knowledge graph.

We are actively training CyBERT with increased compute and building larger NER training datasets. The CyBERT model versions are publicly available on the CyBERT Github repository: https://github.com/priyankaranade1/CyBERT

## References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[3] Wikipedia. Wikipedia corpus. https://dumps.wikimedia.org/.

[4] W.N. Francis and H. Kucera. The brown corpus of standard american english. https://archive.org/details/BrownCorpus.

[5] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv:1906.01787*, 2019.

[6] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing*, pages 3721–3731. ACL, 2019.

[7] Taihua Shao, Yupu Guo, Honghui Chen, and Zepeng Hao. Transformer-based neural network for answer selection in question answering. *IEEE Access*, 7:26146–26156, 2019.

[8] Hugging Face. Pretrained models. https://huggingface.co/transformers/pretrained_models.html.

[9] Google Research. Understanding searches better than ever before. https://blog.google/products/search/search-language-understanding-bert/.

[10] Jieh-Sheng Lee and Jieh Hsiang. Patent claim generation by fine-tuning OpenAI GPT-2. *arXiv:1907.02052*, 2019.

[11] Steven Y Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. Genaug: Data augmentation for finetuning text generators. In *Deep Learning Inside Out: 1st Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42, 2020.

[12] Priyanka Ranade, Aritran Piplai, Sudip Mittal, Anupam Joshi, and Tim Finin. Generating fake cyber threat intelligence using transformer-based models. In *International Joint Conference on Neural Networks 2021 (IJCNN 2021)*, 2021.

[13] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[14] Nitika Khurana, Sudip Mittal, Aritran Piplai, and Anupam Joshi. Preventing poisoning attacks on ai based threat intelligence systems. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

[15] Priyanka Ranade, Sudip Mittal, Anupam Joshi, and Karuna Joshi. Using deep neural networks to translate multi-lingual threat intelligence. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 238–243. IEEE, 2018.

[16] Sudip Mittal, Anupam Joshi, and Tim Finin. Cyber-all-intel: An ai for security related threat intelligence. *UMBC Faculty Collection*, 2019.

[17] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019.

[18] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating cybersecurity knowledge graphs from malware after action reports. *IEEE Access*, 2020.

[19] Sudip Mittal, Prajit Das, Varish Mulwad, Anupam Joshi, and Tim Finin. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2016.

[20] M Lisa Mathews, Paul Halvorsen, Anupam Joshi, and Tim Finin. A collaborative approach to situational awareness for cybersecurity. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 216–222. IEEE, 2012.

[21] Ouissem Ben Fredj, Alaeddine Mihoub, Moez Krichen, Omar Cheikhrouhou, and Abdelouahid Derhab. Cybersecurity attack prediction: a deep learning approach. In *13th International Conference on Security of Information and Networks*, pages 1–6, 2020.

[22] Hiromu Yakura, Shinnosuke Shinozaki, Reon Nishimura, Yoshihiro Oyama, and Jun Sakuma. Malware analysis of imaged binary samples by convolutional neural network with attention mechanism. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pages 127–134, 2018.

[23] Lorenzo Neil, Sudip Mittal, and Anupam Joshi. Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports. *IEEE Intelligence and Security Informatics (IEEE ISI) 2018*, 2018.

[24] Ketki Sane, Karuna Pande Joshi, and Sudip Mittal. Semantically rich framework to automate cyber insurance services. *IEEE Transactions on Services Computing*, 2021.

[25] Matthew Sills, Priyanka Ranade, and Sudip Mittal. Cybersecurity threat intelligence augmentation and embedding improvement-a healthcare usecase. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2020.

[26] Hanna M Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984, 2006.

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[28] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[29] Liang Yao, Chengsheng Mao, and Yuan Luo. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*, 2019.

[30] Brian Krebs. Krebs on security. https://krebsonsecurity.com/, 2021.

[31] Harold Booth, Doug Rike, and Gregory Witte. The national vulnerability database (nvd): Overview. Technical report, National Institute of Standards and Technology, 2013.

[32] Mitre Corporation. Common vulnerabilities & exploitations. https://cve.mitre.org/.

[33] aptnotes. APTnotes repository. https://github.com/aptnotes/data, 2021.

[34] Leyla Bilge and Tudor Dumitraş. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844, 2012.

[35] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[36] Wen Tai, HT Kung, Xin Luna Dong, Marcus Comiter, and Chang-Fu Kuo. exbert: Extending pre-trained models with domain-specific vocabulary under constrained training resources. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1433–1439, 2020.

[37] spaCy. Industrial Strength Natural Language Processing. https://spacy.io/.

[38] ”HuggingFace”. ”https://github.com/huggingface”.

[39] Shodan. Search Engine for the Internet of Everything. https://www.shodan.io/.

[40] Stanisław Saganowski. Cybersecurity NER corpus 2019, 2020.

[41] Soham Dasgupta, Aritran Piplai, Anantaa Kotal, and Anupam Joshi. A comparative study of deep learning based named entity recognition algorithms for cybersecurity. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2596–2604. IEEE, 2020.

[42] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun’ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, 2012.

[43] microsoft. Security Bulletins. https://docs.microsoft.com/en-us/security-updates/securitybulletins/securitybulletins.

[44] adobe. Security Bulletins and Advisories. https://helpx.adobe.com/security/security-bulletin.html.

[45] NIST. After Action Reports. https://csrc.nist.gov/glossary/term/after_action_report.