A Comparative Study of Deep Learning based Named Entity Recognition Algorithms for Cybersecurity

Soham Dasgupta†, Aritran Piplai*, Anantaa Kotal*, Anupam Joshi*
*Dept. of Computer Science & Electrical Engineering, University of Maryland, Baltimore County,
Email: {apiplai1,anantak1, joshi}@umbc.edu
†Mallya Aditi International School, Email: sohamdasgupta91@gmail.com

***Abstract*—Named Entity Recognition (NER) is important in the cybersecurity domain. It helps researchers extract cyber threat information from unstructured text sources. The extracted cyber-entities or key expressions can be used to model a cyber-attack described in an open-source text. A large number of general-purpose NER algorithms have been published that work well in text analysis. These algorithms do not perform well when applied to the cybersecurity domain. In the field of cybersecurity, the open-source text available varies greatly in complexity and underlying structure of the sentences. General-purpose NER algorithms can misrepresent domain-specific words, such as "malicious" and "javascript". In this paper, we compare the recent deep learning-based NER algorithms on a cybersecurity dataset. We created a cybersecurity dataset collected from various sources, including "Microsoft Security Bulletin" and "Adobe Security Updates". Some of these approaches proposed in literature were not used for Cybersecurity. Others are innovations proposed by us. This comparative study helps us identify the NER algorithms that are robust and can work well in sentences taken from a large number of cybersecurity sources. We tabulate their performance on the test set and identify the best NER algorithm for a cybersecurity corpus. We also discuss the different embedding strategies that aid in the process of NER for the chosen deep learning algorithms.**

***Index Terms*—Named Entity Recognition, Deep Learning, Cybersecurity, Artificial Intelligence**

I. Introduction

Cyber-threat intelligence (CTI) provides security researchers with knowledge that helps them to detect and understand a cyber-attack, and also ways to mitigate the attack. Security researchers often detonate a malware sample and publish detailed incisive reports about the behavior of the malware and ways to mitigate it. Software companies also report security threats that affect the products that they develop, usually on their respective websites. They also mention different patches, if available, that may eradicate the security threats affecting their products. There are also multiple social media feeds that talk about cyber-attacks and their characteristics. This provides a large repertoire of information that is available in natural language about various cyber-attacks, making it possible for researchers to mine CTI from these sources. One of the ways [25] to extract CTI is to use automated, often machine learning-based, algorithms to mine knowledge from these open-source data feeds about cyber-attacks.

To automatically pick out portions of the data feeds that might be of interest to security researchers, we need an algorithm to map specific words, or a group of words, to some known classes which are necessary to map out (describe) a cyber-attack. NER helps in this part of the exercise, as it takes a piece of text as input and maps a sequence of words to some pre-defined classes and also marks the irrelevant pieces of information. It can then be used as a part of a pipeline which does threat extraction and represents the extracted information in a knowledge graph [25], [31], [33], for example, one defined by the Unified Cybersecurity Ontology [41].

Although NER algorithms have existed for a significant period of time, recent developments in neural networks and deep learning have outperformed the previous NER algorithms. We focus only on the deep learning-based algorithms in this paper, and we compare all the algorithms to a baseline of Stanford NER - a CRF based NER which has also been used in the cybersecurity domain [14], [28].

There are various general-purpose NER algorithms, and there has been significant research conducted in this area. However, when the same algorithm is applied in the cybersecurity domain we are not guaranteed to reach the same outcome. Recent developments in deep learning have made it possible for an algorithm trained in one domain, to be used in another domain. However, as Anastasiia *et al.* [36] state, for NER this method is not guaranteed to be successful. For domain-specific NER, we have to create our own training dataset and annotate the dataset with proper labels for machine learning-based NER techniques. The structure of the sentences can greatly vary depending upon the depth of technicality present in the source of information, even within the domain. For example, an NER algorithm mining information from a Twitter feed about a cyber-attack may not always work for mining information from a detailed technical report about the same cyber-attack. A Twitter user may want to squeeze in a lot of information in their feed due to the limitations set by the social media platform. In contrast, a detailed technical report may span out the knowledge into multiple pages, greatly varying the sentence structure rendering the machine learning algorithm

This work supported in part by an award from DoD to Joshi.

confused. Similarly some of the well-known databases which store information about vulnerabilities and possible cyber-attacks that might take advantage of these vulnerabilities, often follow a rough template while describing them. Although these pieces of information are still unstructured, a machine learning-based algorithm trained to detect NEs may perform well on a test set which solely consists of text from the sources which follow a rough template while describing cyber-attacks. However, this may not necessarily mean that the NER is performing well for text from any source. Hence, it is important that we consider the text from a wide variety of heterogeneous sources, and further study the performance of different NER algorithms on an appropriate test set that includes text from multiple sources.

In this paper, we present a NER dataset which we created by carefully curating data feeds from a wide range of sources. We have sentences from short feeds about vulnerabilities and threats like MSB, [20] Adobe Security. [1] We also incorporated sentences from long technical reports. We subsequently test the performance of a list of recent neural network-based NER algorithms and tabulate the results. Since we focus on deep learning-based algorithms for NER in cybersecurity, we further discuss the important topic of embeddings of the individual words, as it applies to natural language understanding. Embeddings are an important part of deep learning algorithms focused on natural language processing. Some deep learning models develop their own embeddings during the training phase, and in some cases, the embeddings are pre-trained from a separate model. We use different combinations of embeddings and deep learning algorithms on the curated dataset. Understanding the word embeddings and visualizing them shed light on the behavior of the underlying models in a given domain - in this case for NER in cybersecurity applications. To this end, we visualize the embeddings and cluster them. We seed the cluster centers based on randomly chosen keywords to visualize how close other entities are in the embedding space. This provides us with an intuition about how different words are being classified into specific entity classes by the neural networks. We believe that the performance comparison of different algorithms on the curated dataset along with the visualization of the embeddings resulting from training is going to be particularly useful for future researchers, who are interested in creating new CTI extraction algorithms with the help of NER.

The remainder of the paper is organized as follows - Section II,describes the related work. We describe the different variations of NER for cybersecurity and dataset for evaluation in Section III. Finally, we describe and compare the different NER algorithms in Section IV and conclude our findings in Section VI.

II. Related Work

In this section, we discuss the previous research conducted to develop NER and also how NER has been used previously for CTI Extraction. In an NER algorithm, we send a sequence of words as an input. The output of the NER algorithm is a label, signifying an entity type, corresponding to each of the words sent as input.

A. Feature based unsupervised NER

Unsupervised NER has been used previously when there is a dearth of annotated data. There have been rule-based unsupervised strategies for NER as is demonstrated by Etzioni et. al. [6] The authors claim that this strategy is mostly domain-independent. However, practitioners need to supply some domain-specific predicate rules for this algorithm to work on a particular domain. With the help of text-based feature extraction, researchers have also developed clustering-based NER algorithms. [29] Recently, instead of text-based features, researchers have also used word embeddings for clustering. [2] Knowledge-base aided strategies are also being used [17] for unsupervised NER. Researchers have claimed to not have to depend on auxiliary information about a candidate entity from the text, but can retrieve features for the same from knowledge bases about the same domain. This approach is interesting, but in the domain of cybersecurity we often need NER to populate a knowledge base. [33]

B. Supervised and Semi-supervised NER

Most of the papers published for NER have used supervised learning for NER detection. Recently Liang et al. in their paper [16] talked about the problems with depending upon distant supervision based on external knowledge bases. They describe a multi-stage general-purpose NER which uses BERT [5] to classify entity based just on the distant labels. After that, they remove the distant labels and self-train the model to achieve better performance. Since the cost of human annotations is high, researchers have been recently invested in semi-supervised approaches for NER detection. Pattern-based bootstrapping methods have been used for NER detection in Tamil and English languages [34]. The researchers have taken a small annotated training set and identified patterns from the context of those annotations. They have subsequently bootstrapped those patterns and used them to classify the entities of the test set. Yang et al. in their paper [39] have described a reinforcement learning-based algorithm that reduces the problem of human annotations. The authors of this paper mainly talk about tackling noisy and incomplete annotations. Their reinforcement learning-based instance selector helps in generating annotations to automatically fill incomplete annotations. Zafarian et al. in their paper [40], have used a semi-supervised strategy to detect NER in the presence of weak labels. They have essentially built a CRF based supervised classifier, and then subsequently used a graph-based classifier that essentially augments the supervised training set, by automatically annotating them.

C. NER for CTI Extraction

NER helps to extract important information from textual data feeds in the domain of cybersecurity. NER has been used to extract vulnerability alerts [7]. In our paper we have

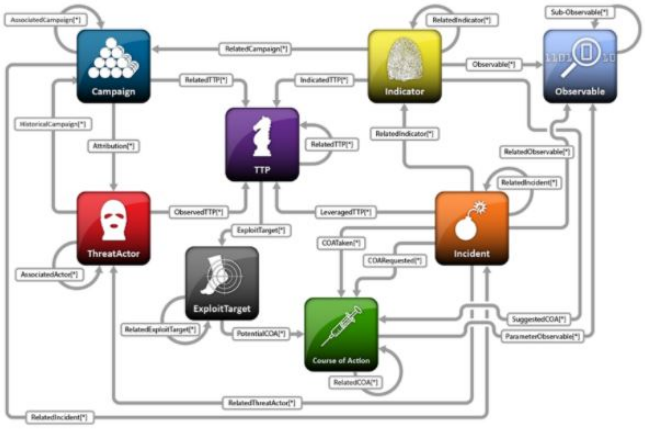

Fig. 1. Overview Structured Threat Information eXpression (STIX) 2.0 architecture [8].

importantly incorporated Common Vulnerabilities Emuneration (CVE [23]) feeds into our dataset. Piplai et. al. in their paper [33] describe a pipeline to represent and model CTI. They have used Stanford NER and Regular Expressions to detect cyber-entities from open source text. Stanford NER [18] has been used previously for NER in multiple CTI pipeline [14]. Mittal et. al in their paper [24], have also used NER to automatically generate alerts from Twitter feeds relevant to cybersecurity. CTI extraction with the help of NER has also been combined with vector spaces to better represent the data [26]. Recently with the advancements of deep learning, LSTM based NERs have been popularized in cybersecurity entity extraction. [11] Some researchers have applied machine learning for NER in cybersecurity, but exclusively for Russian language. In this paper [27], the authors have proposed a two stage NER pipeline which collects features from the text as well as the whole document. Recently, BERT has been used for cyber NER in the Russian domain. [37] This paper compares these models on our dataset, which is in English language.

D. Previous Surveys and Studies on NER

There is a comprehensive survey on domain-independent NER available. [15] This study is quite extensive. However, this paper does not talk about a specific domain, and does not compare the performance of the different algorithms on a carefully curated data set. Domain specific NER studies have been published for the medical domain. [13], [38] A cybersecurity specific NER study is also available which focuses on NER algorithms for cybersecurity in Russian language. [19] This paper talks about the different NER algorithms for the information security domain. Although this paper is quite interesting, it fails to compare the 'attention' based NLP algorithms (like BERT [5]) that have subsequently been used in NER in the cybersecurity domain.

III. Methodology

In this section, we discuss the different NER algorithms that we have chosen for comparison. We also talk about the dataset

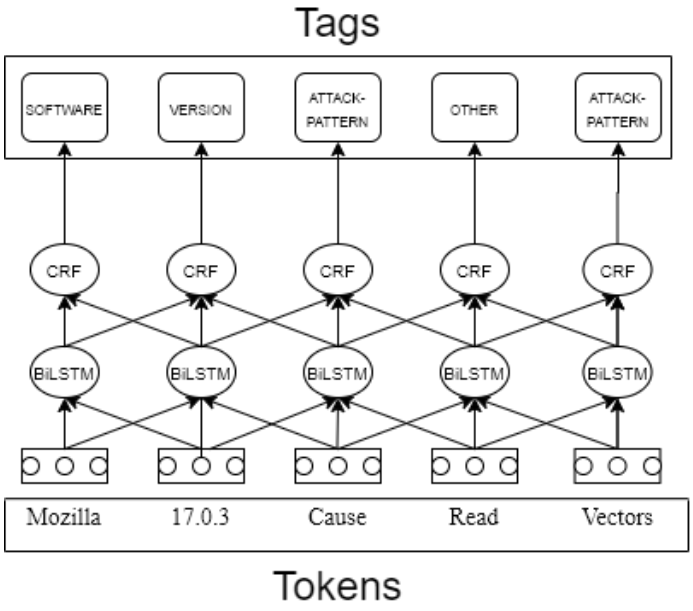

Fig. 2. BiLSTM + BERT on a part of a sentence from the test set

for training and testing, and the NER classes we aim to map the entities into.

A. Dataset collection and Description

General purpose NERs do not necessarily work well for a particular domain. In some cases, it is imperative that we build a dataset on our own if we want to build an NER for a particular domain. Most of the prior work on NER for cybersecurity has concentrated on CVE and NVD feeds for their data. Annotated datasets required for supervised learning are not readily available for cybersecurity. Even if an annotated dataset is available, it is difficult to reach a consensus about what classes need to be extracted from the data. Recently an NER corpus for cybersecurity has become available for researchers to use [35]. But even this dataset has concentrated only on twitter feeds, and annotated the feeds with a few classes that may not necessarily help cyber-researchers to extract meaningful information. In order to compare different NER algorithms, we need to test them on a dataset that has a mix of different types of feeds. We create a dataset of 2100 sentences collected from various sources.

- Microsoft Security Bulletin [20]
- Adobe Security Updates [1]
- Twitter feeds
- Long Technical Reports
- Common Vulnerabilities Enumeration [22]
- National Vulnerabilities Database [30]

These sources provide a mixed bag of sentences which is ideal for testing out different algorithms for NER. Some NER algorithms trained on CVE and NVD sentences may not work well for long technical reports. The information in sentences from small pieces of text is often condensed, compared to

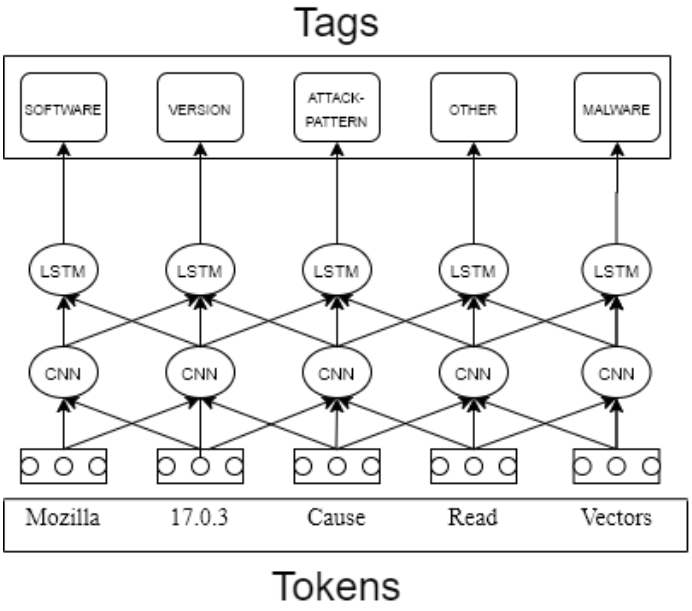

Fig. 3. BERT + CNN + LSTM on a part of a sentence from the test set

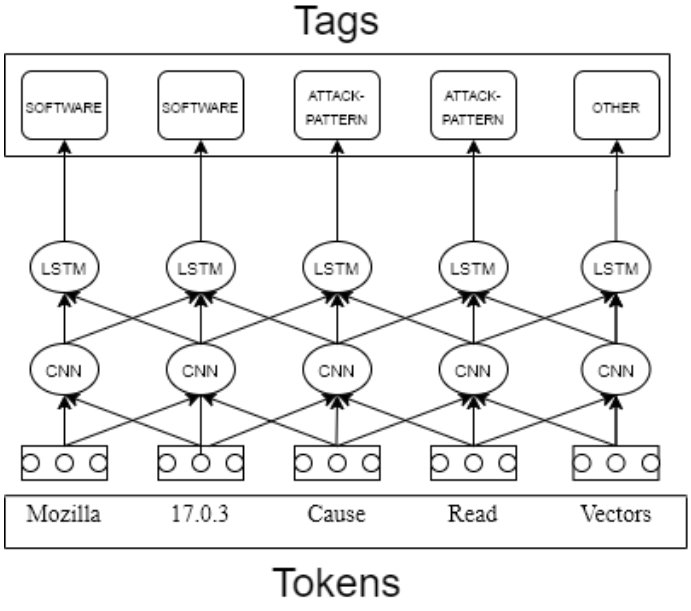

Fig. 5. Word2Vec + CNN + LSTM on a part of a sentence from the test set

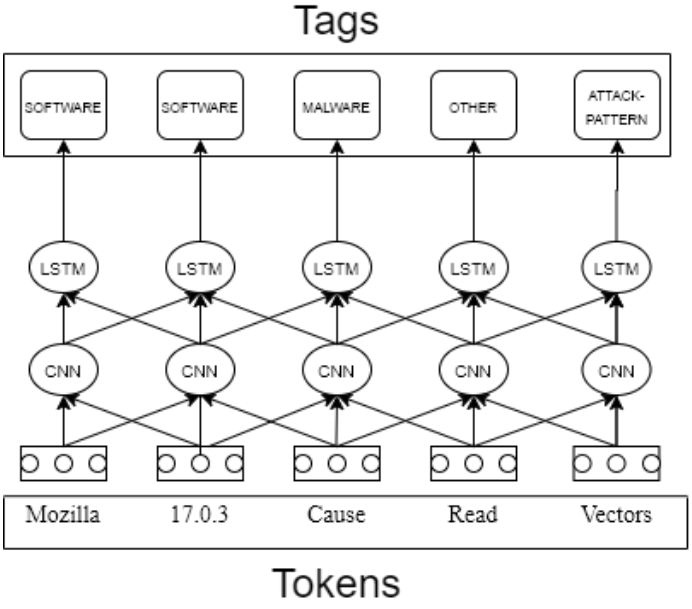

Fig. 4. UMBC Word2Vec + CNN + LSTM on a part of a sentence from the test set

sparse descriptive reports about cyber-attacks. We have curated 550 pieces of text from Microsoft and Adobe and we have 474 Technical Reports about cybersecurity.

B. Classes

Next we describe the classes that we map our entities into. In order to extract meaningful CTI, we need to create entity classes which help in modeling a cyber-attack. Structured Threat Information eXpression (STIX) is a standard for exchanging threat intelligence information. It is very detailed and helps to accurately represent a cyber-attack. The base schema for STIX 1.0 [9] has been upgraded to STIX 2.0 [8]. Figure 1 represents an overview of the groups of classes used in STIX. A Unified Cybersecurity Ontology [41] (UCO) has been developed based on the STIX format. We use a subset of the classes mentioned in UCO to form our entity classes. We do not use all the classes from UCO, as classes like 'IP Addresses', 'SHA256 Hashes' are detected by regular expressions with a much greater accuracy. These pieces of texts belonging to such classes do not need any contextual information in order to make a prediction. We have also added a class called 'Exploit-Target' which helps in representing the specifications of the site of the attack.

Here we explain some important classes that are present in UCO 2.0 as mentioned in [31]–[33]:

- *Software* : An entity that relates to a piece of code such as a product of a company that may contain a 'Vulnerability' or can be used as a 'Tool'.
- *Version* : An entity related to the entity class 'Software' that describes the category of the 'Software' published.
- *Malware* : An entity that refers to malicious code carrying out an attack in a system or an 'Exploit-Target'.
- *Vulnerability* : An entity that means a weakness that could be exploited by adversaries.
- *Course-of-action* : An entity that refers a sequence of actions that either prevents or mitigates an attack.
- *Tool* : An entity that refers to a legitimate software or a piece of code that can be used by adversaries for malicious activities.
- *Attack-pattern* : An entity that refers to a sequence of actions that could result in a cyber-attack.

C. Models

We choose from a wide variety of neural network architectures proven to perform well on NER tasks. In specific, we describe three models which have demonstrated higher performance when evaluated on the curated dataset.

1) LSTM + CRF: Huang et al. first published in their paper [12] the use of LSTMs and CRFs for sequence tagging or NER. Long-Short Term Memory(LSTMs) are an improvement over Recurrent Neural Networks (RNNs). RNNs, for a sequence tagging task, take as an input a vectorized form of the input words. The vector can be a one-hot encoding or an embedding feature. A hidden layer at time 't' carries information from the hidden layer at time 't-1', with no restriction. On theory, unrestricted information from the word's history is captured in this hidden cell. The final output layer is usually a softmax layer that has the dimension of the total number of classes. For NER, this would be the number of entity-types. However, RNNs face common neural network challenges like vanishing and exploding gradient. It may also suffer from being unable to distinguish between recent and obsolete history. LSTMs address these problems by introducing different gates that aim to 'retain' or 'forget' parts of the history. The gates have individual activation functions and individual weights. Conditional Random Fields (CRFs) perform well for an additional post-processing step on the models. What CRFs essentially do is that they perform an additional feature extraction step on the sentence level. They act on the last hidden layer of the LSTM model, create a feature extraction pipeline and output the softmax values at the final layer. In our experiments, we have 100 LSTM cells, and a Time distributed layer.

2) Bi-directional LSTMs + CRF: In the same paper that discussed LSTMs and CRFs for NER, [12] the authors have also stated the use of BiDirectional LSTMs with CRFs for the same purpose. BiDirectional LSTMs are an improvement over LSTMs and perform better than LSTMs in some specific cases. BiDirectional LSTMs, similar to LSTMs, work on capturing the history of a particular word with the help of its hidden cells. It also captures the future of the word in a second pass. For every hidden layer cell capturing the history of a particular word in a sentence, there is another hidden layer cell which essentially takes the help of the future of the word in that given sentence. This is very useful for the hidden layer representations of certain words, for tasks like sequence tagging, because the 'classification triggers' may not always be present in the part of the sentence preceding a particular word. Sometimes the trigger can be present in the part of the sentence succeeding the same word. The same concept of gates for retaining or disregarding part of the history or future also exists for BiLSTMs. CRFs are also used as a part of the post-processing step. CRFs work on the hidden layer representations of each sentence, generate additional features and conclude the classification step with the softmax layer that generates probabilities for different entity classes for each word. In our experiments, we have used 100 BiLSTM cells, and a Time distributed layer.

3) LSTMs +CNN: Nichols et al. [4] described a method of using LSTMs and Convolutional Neural Networks for NER. As described in the models before, the LSTM captures the history of a particular word in its hidden units. The fundamental difference between this model and the previous models is that the post-processing step of CRF is replaced with the pre-processing step of CNN. In the other models, CRF is used to extract additional features from the sentence level. The hidden layer representations of the words along with the entire sentence are sent as input to the CRF model. In this particular model, CNN is used to extract character level features for each word. This is the only model that uses the structure of the word as an input to the LSTM model. This is useful, especially for cybersecurity domain, because the structure of the words does play an important role in deciding the entity type of some classes. For example 'CVE-1216' and 'CVE-1217' are both vulnerabilities. In this case, the structure of the word is more important than the context for deciding the entity class of the particular word.

D. Embeddings

Embeddings are an important part of deep learning models for sequence tagging tasks. The representations for each word place it in a vector space that helps the neural network to capture additional information about the context.

1) Word2Vec: Word2Vec [21] has been one of the most famous embedding strategies after deep learning algorithms became popularized. Word2vec can be used to vectorize each word of a piece of text, and the word vectors are claimed to be related to each other semantically. In Section **??** we will see examples of how embeddings of different entities form clusters. Word2Vec uses a Continuous Bag of Words model as an input. It uses the context of a current word, multiplies it with a matrix, and tries to predict the current word. The second model uses the word as an input and tries to regenerate the context of the current word based on the input word. This model is called the continuous skip-gram model. The Word2Vec algorithm aims at learning representations which will best help in predicting the context of the word for which the representation is being learnt. In our experiments, we used a pre-trained general-purpose Word2Vec. We also used a pre-trained cybersecurity Word2Vec embedding model. The general purpose Word2Vec embedding model that we have used in our embedding is Word2Vec model [10] trained on a Google News corpus. The cybersecurity embedding model that we have used for our experiments is a Word2Vec model trained on a cybersecurity corpus [3]. There is not any overlap with the dataset that we are using for training the neural network model and testing it.

2) BERT: Bidirectional Encoder Representations from Transformers (BERT) [5] has become popular lately along with other transformer architectures. BERT claims to be better than Word2Vec at generating word embeddings because it creates contextualized embeddings for a single word as opposed to a single embedding generated by Word2Vec. This

TABLE I
COMPARISON OF RESULTS ON DATASET

Test and Validation Results					
Method	**Test Acc.**	**Validation Acc.**	**Precision.**	**Recall.**	**F-1.**
Stanford NER	91.6	90.8	78.00	78.00	77.00
Domain independent Word2vec+LSTM +CRF	96.96	96.71	85.00	77.00	81.00
Domain independent Word2vec+BiLSTM+CRF	97.40	97.09	88.00	80.00	84.10
Domain independent Word2vec+CNN+LSTM	97.59	96.64	84.00	80.00	82.00
Domain-specific Word2vec+LSTM +CRF	97.31	96.16	90.00	76.00	82.40
Domain-specific Word2vec+BiLSTM+CRF	97.20	96.80	88.00	81.00	84.30
Domain-specific+CNN+LSTM	97.66	97.18	90.00	81.00	85.20
BERT+LSTM+CRF	97.73	96.93	90.00	83.00	86.20
BERT+BiLSTM+CRF	98.10	97.18	93.00	84.60	88.60
BERT+CNN+LSTM	97.50	96.80	91.00	80.30	85.30

[a]Sample of a Table footnote.

TABLE II
COMPARISON OF 10 FOLD CROSS VALIDATION RESULTS OF THE MODELS

Training Results		
Method	**Train Acc.**	**10 fold cross validation**
Word2vec+LSTM +CRF	97.92	95.42
Word2vec+BiLSTM+CRF	97.72	95.29
Word2vec+CNN+LSTM	98.10	95.25
UMBC Word2vec+LSTM +CRF	97.48	95.41
UMBC Word2vec+BiLSTM+CRF	98.70	95.55
UMBC Word2vec+CNN+LSTM	99.18	95.94
BERT+LSTM+CRF	97.76	95.45
BERT+BiLSTM+CRF	98.23	95.26
BERT+CNN+LSTM	98.68	95.22

[a]Sample of a Table footnote.

might be helpful across multiple domains. In the domain of cybersecurity, the embedding for the word 'Internet' should be different if it is followed by the word 'Explorer' than any other word. Although Word2Vec considers context before generating an embedding, it uses all the possible contextual words that can be present for a particular word for training and generates a single embedding for that word. BERT takes context into account, even while generating embeddings. BERT achieves this by masking some of the words in a particular sentence. The aim of the algorithm is to generate the masked words from the context. The second objective of the algorithm is 'Next Sentence Prediction'. In this phase, a sentence followed by its succeeding sentence in the corpus is provided as input. We also need to provide two randomly picked sentences from the corpus as input, such that they are not successive sentences in the corpus. The objective of the algorithm is to predict the next sentence from the data. This particular strategy may help in generating embeddings for cyber-entities from long technical reports. BERT has a lot of capabilities. One particular case where BERT works well is a 'question-answering' task. In our experiment, we extract the embedding layer of BERT and use it as an input for different deep learning-based algorithms. Since BERT requires a huge corpus for training, we use the pre-trained model and fine-tune it with our cybersecurity corpus. A pre-trained model is augmented with some of the sentences from our training set to improve the embeddings of cyber-security embeddings. We use about 1500 sentences from the training set and supply that to the pre-trained BERT model for fine-tuning. Then we extract the embedding from the fine-tuned BERT model and use it.

IV. EXPERIMENTS AND RESULTS

The training set we used constitutes of 1618 sentences. The model specifications have been summarized below. We can see the detailed evaluations of all the models in Table I. Table II shows the 10 fold cross validation result of the different models.

A. BERT+BiLSTM

The input layer is a 768 dimensional embedding layer, that we extracted from a fine-tuned BERT model. The neural network consists of 128 BiLSTM cells. It is followed by a Time Distributed layer of 50 neurons, and a subsequent softmax layer of 9 possible classes. We observe that BERT embeddings with a BiLSTM model, along with a CRF post-processing layer results in a test accuracy of 98.10 %. We use F-1 score as the key metric to compare the different models. We observe that BERT+ BiLSTM results in the highest F-1 score of 88.60 %.

B. BERT + LSTM

We use the same 768 dimensional embedding layer for LSTM. The main difference being that in this model we use an LSTM model with 256 LSTM cells. We have another Time Distributed layer of 50 neurons, and a softmax layer of 9 possible classes. In one model we have 5 parallel 1-dimensional CNN models, with 256 filters. We observe that this model results in an F-1 score of 85.30 %. In another model, we use CRF as a post-processing step and this results in a slightly better F-1 score of 86.20 %. This means that BERT

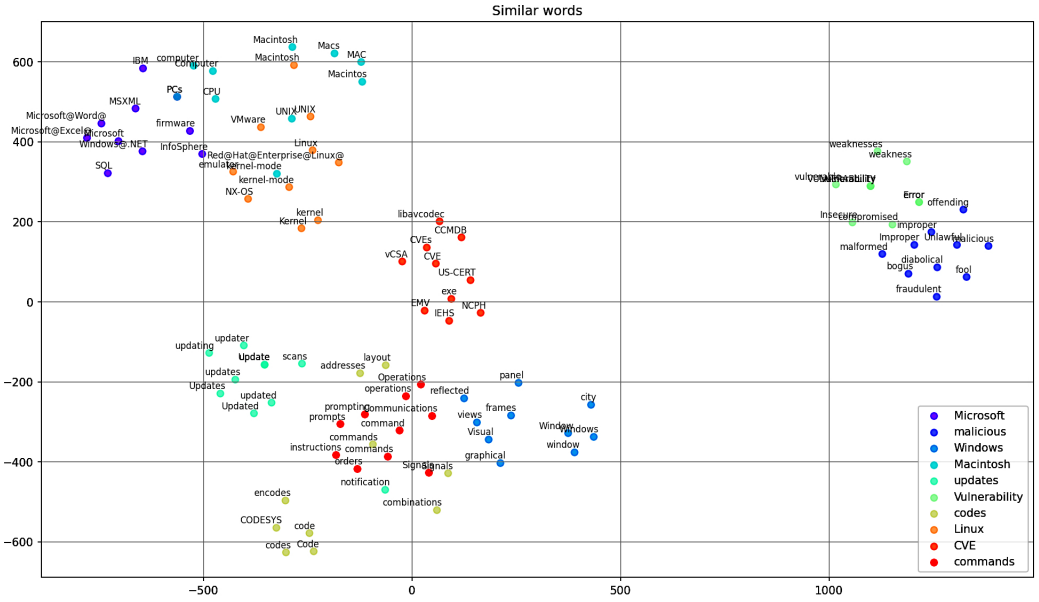

Fig. 6. Clusters of BERT Embeddings

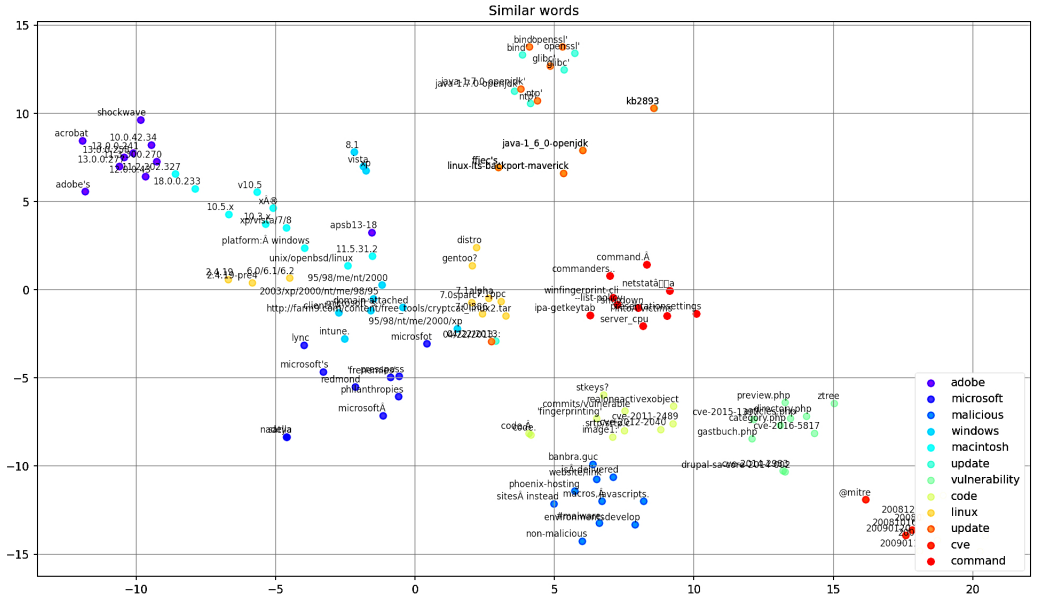

Fig. 7. Clusters of Word2vec Embeddings

has captured the character level distribution efficiently, and an additional CNN layer may not be useful for a model using BERT embeddings. This may be due to BERT's ability to capture 'word inflection'. For example 'Adobe' and 'Adobe's' mostly refer to the same entity type, even though they form separate words in our corpus.

C. Domain independent Word2vec + BiLSTM

In this model we use an external Word2Vec embedding for the BiLSTM model. The Word2Vec model is not specific to the cybersecurity corpus. We include this model in our experiments to understand if domain-independent embeddings can provide us with additional semantic information that may be missing from other embedding models specifically trained on a cybersecurity corpus. The Word2Vec model results in a 300 dimensional embedding for the words in our corpus. The BiLSTM layer consists of 128 BiLSTM cells. We have a similar Time Distributed Layer with 50 neurons, and a softmax layer with 9 classes. We also have a post-processing CRF layer for this model. We observe that this model results in an F-1 score of 84.10 % in the test set.

D. Domain independent Word2vec + LSTM

In these models we use the same external Word2Vec embedding. We use an LSTM model in this case. The input layer is a 300 dimensional embedding layer that we extract from the Word2Vec model. The LSTM model has 256 LSTM cells, and a Time distributed layer consisting of 50 neurons. It is followed by a softmax layer with 9 classes. This architecture results in two separate models. One with a CNN input layer to capture character level information, and the other with a CRF layer for post-processing. The CNN based model consists of 5 parallel 1 dimensional CNN layer with 256 filters. This particular model results in an F-1 score of 82 %. The CRF based model achieves a slightly lower score of 81.00 %. We believe that Word2Vec fails to capture 'word inflection' as efficiently as BERT. This is because Word2Vec generates embeddings only from the context window of a particular word. Hence, a CNN layer to capture character level information might be helpful in detecting certain classes.

E. Domain specific Word2vec + BiLSTM

We have a trained Word2Vec model, specifically for cybersecurity domain. This has been trained on a cybersecurity corpus of 1 million words. We use this model for experiments to check if the Word2Vec embeddings trained for the specific domain are indeed better than domain independent embeddings. The embedding model vectorizes each word into a 100 dimensional vector. We use this embedding in the same BiLSTM model with 128 BiLSTM cells. We add a post-processing step with CRF. This model achieves a higher F-1 score of 84.30 % compared to the F-1 score of the same model with domain-independent Word2Vec embeddings. This means that there is an improvement in the classification with the help of domain specific embeddings.

F. Domain specific Word2vec + LSTM

We use the domain specific 100 dimensional Word2Vec embedding in the LSTM models. Similar to the previous cases, we use two flavors of LSTM models. One with the character level CNN layer, and the other with a CRF based post-processing layer. The CRF based model achieves an F-1 score of 82.40 %. The CNN based model has 5 parallel 1-dimensional CNN layers with 256 filters. This model achieves an F-1 score of 85.20 %. It is interesting to note that both these models with domain specific Word2Vec embedding performed better than their domain-independent counterparts. The CNN based model achieves the best F-1 score out of the 3 models built from this embedding. This may be because Word2Vec fails to capture character level similarity as it is not an implicit part of the algorithm. The CNN layer helps in identifying some of the entities that have important character level features.

We can observe that the models using BERT embeddings, even though not trained entirely on a cybersecurity corpus, achieve better scores than models using Word2Vec embeddings. Among the models using Word2Vec embeddings, we can see that the CNN based models perform slightly better than the CRF based models. The character level understanding is helpful for Word2vec, but since BERT has built-in character level information in its embedding step, it is not necessarily an advantage.

V. Understanding Embeddings

We specify a list of keywords in Cybersecurity for which we want to illustrate the nearest neighbors, as determined by Word2Vec and BERT respectively. For each keyword, we choose 10 closest neighbors to form a neighborhood cluster. For each keyword, we generate neighborhood clusters from Word2Vec and BERT. For the Word2Vec embeddings, we use the most_similar attribute of the Word2Vec model to create a neighborhood cluster of 10 neighbors for each keyword.

For the BERT embeddings, the nearest neighbor for each keyword is determined by measuring the cosine distance between the keyword embedding and embeddings for the other words in our training set. We choose the top 10 words with BERT embeddings closest to our keyword.

We want to illustrate the neighborhood clusters drawn from each embedding to point out the differences. To visualise our neighborhood clusters, we used t-SNE embedding. It uses stochastic neighbor embedding to preserve neighborhood identities while embedding our high-dimensional data in two-dimensions for the purpose of visualization. Even though the dimensionality has been reduced, the neighborhood properties are preserved. This allows us to make conclusions about similar words in our dataset. We used the TSNE in class from the 'sklearn.manifold' module in python for our experiments. The word clusters generated from BERT and Word2Vec Embeddings are shown in Figure 6 and 7. Since we fine-tune the BERT model and we use a Word2vec model trained on a cybersecurity corpus, we observe that the clusters formed for the Word2vec model are consistent with cybersecurity terms. The BERT model captures a mixture of general terms and cybersecurity terms. However, this may be beneficial for a deep learning model as it requires information related to cybersecurity, as well as general information to reach a conclusion about the category of a given word.

VI. Conclusion and Future Work

We successfully compared the recent deep learning-based algorithms that have been applied to cybersecurity NER. We used some domain-independent embeddings and domain-specific embeddings and concluded that domain-dependent embeddings result in better scores for the same deep learning model. We also inferred that a BERT model fine-tuned with a small cybersecurity corpus can yield better results than a model using Word2vec embeddings trained on a huge cybersecurity corpus. Usually, in comparative studies for cybersecurity NER research have focused on a specific data feed that usually has some underlying structural similarity. However, we trained and applied the models on a carefully curated dataset that includes a variety of long and short texts, with varying complexities in the sentence structures. We also use the entity classes specified in STIX, which is an industry standard for exchanging threat intelligence information. In

the future, we would like to include more algorithms for cyber NER comparison. With the recent revival of knowledge-grounded AI algorithms, and as cybersecurity knowledge bases are beginning to gain traction, we can include knowledge guided cyber NER, and see how it fares compared to the recent deep learning algorithms for the same purpose.

REFERENCES

[1] Adobe. Adobe security bulletin. https://helpx.adobe.com/security.html.

[2] Abdulkareem Alsudais and Hovig Tchalian. Clustering prominent named entities in topic-specific text corpora. In *25th Americas Conference on Information Systems, AMCIS 2019, Cancún, Mexico, August 15-17, 2019*. Association for Information Systems, 2019.

[3] Taneeya W. Satyapanich Francis Ferraro Shimei Pan Youngja Park Anupam Joshi Ankur Padia, Arpita Roy and Tim Finin. UMBC at SemEval-2018 Task 8: Understanding Text about Malware. pages 878–884.

[4] Jason P.C. Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016.

[5] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[6] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, S. Soderland, Daniel S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165:91–134, 2005.

[7] TM Georgescu and Zurini M. Iancu B. Named-entity-recognition-based automated system for diagnosing cybersecurity situations in iot networks. In *PubMed Sensors (Basel, Switzerland)*. MDPI, 2020.

[8] Oasis group. Stix 2.0 documentation. https://oasis-open.github.io/cti-documentation/stix/examples.html, May 2013.

[9] Oasis group. Stix 1.0 documentation. https://stixproject.github.io/documentation/, May 2018.

[10] MMI Haltz. Google news vectors. https://github.com/mmihaltz/word2vec-GoogleNews-vectors, May 2015.

[11] Jannik Laval Houssem Gasmi and Abdelaziz Bouras2. Information extraction of cybersecurity concepts:an lstm approach. In *Applied Sciences 9(19):3945*. MDPI, 2019.

[12] Zhiheng Huang, W. Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *ArXiv*, abs/1508.01991, 2015.

[13] Lei J, Tang B, Lu X, Gao K, Jiang M, Xu H., and J Am. A comprehensive study of named entity recognition in chinese clinical text. *Med Inform Assoc.*, 2014.

[14] Ravendar Lal. Information Extraction of Security related entities and concepts from unstructured text. Master's thesis, May 2013.

[15] J. Li, A. Sun, J. Han, and C. Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[16] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, page 1054–1064, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Angli Liu, Jingfei Du, and Veselin Stoyanov. Knowledge-augmented language model and its application to unsupervised named-entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1142–1150, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[18] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Prismatic Inc, Steven J. Bethard, and David Mcclosky. The stanford corenlp natural language processing toolkit. In *In ACL, System Demonstrations*, 2014.

[19] Ivan Mazharov and Boris V. Dobrov. Named entity recognition for information security domain. In *DAMDID/RCDL*, 2018.

[20] Microsoft. Microsoft security bulletin. https://msrc-blog.microsoft.com/tag/security-bulletin/.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[22] MITRE. Cvelist project. https://github.com/CVEProject/cvelist.

[23] MITRE. Cvelist project. https://github.com/CVEProject/cvelist, May 2013.

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

[25] Sudip Mittal, Prajit Kumar Das, Varish Mulwad, Anupam Joshi, and Tim Finin. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 860–867. IEEE Press, 2016.

[26] Sudip Mittal, Anupam Joshi, and Tim Finin. Thinking, fast and slow: Combining vector spaces and knowledge graphs. *arXiv preprint arXiv:1708.03310*, 2017.

[27] V. Mozharova and N. Loukachevitch. Two-stage approach in russian named entity recognition. In *2016 International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT)*, pages 1–6, 2016.

[28] Varish Mulwad, Wenjia Li, Anupam Joshi, Tim Finin, and Krishnamurthy Viswanathan. Extracting information about security vulnerabilities from web text. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '11, pages 257–260, Washington, DC, USA, 2011. IEEE Computer Society.

[29] David Nadeu and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticæ Investigationes*, 2006.

[30] NisT. Nvd. https://nvd.nist.gov/.

[31] 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. In *Int. Conf. on Advances in Social Networks Analysis and Mining*. IEEE, 2019.

[32] Aritran Piplai, Sudip Mittal, Mahmoud Abdelsalam, Maanak Gupta, Anupam Joshi, and Tim Finin. Knowledge enrichment by fusing representations for malware threat intelligence and behavior. *IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2020.

[33] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating cybersecurity knowledge graphs from malware after action reports. *UMBC Faculty Collection*, 2019.

[34] J. Balaji S. Thenmalar and T.V. Geetha. Semi-supervised bootstrapping approach for named entity recognition. In *International Journal on Natural Language Computing*. IEEE, 2015.

[35] Stanisław Saganowski. Cybersecurity NER corpus 2019. 2020.

[36] Anastasiia Sirotina and Natalia Loukachevich. Named entity recognition in information security domain for russian. In *Proceedings of Recent Advances in Natural Language Processing*. ACL, 2019.

[37] Mikhail Tikhomirov, N. Loukachevitch, Anastasiia Sirotina, and Boris Dobrov. Using bert and augmentation in named entity recognition for cybersecurity domain. In Elisabeth Métais, Farid Meziane, Helmut Horacek, and Philipp Cimiano, editors, *Natural Language Processing and Information Systems*, pages 16–24, Cham, 2020. Springer International Publishing.

[38] Wu Y, Jiang M, Xu J, Zhi D, and Xu H. Clinical named entity recognition using deep learning models. *AMIA Annu Symp Proc. 2018;2017:1812-1819.*, Apr 16, 2018.

[39] Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. Distantly supervised NER with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2159–2169, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[40] A. Zafarian, A. Rokni, S. Khadivi, and S. Ghiasifard. Semi-supervised learning for named entity recognition using weakly labeled training data. In *2015 The International Symposium on Artificial Intelligence and Signal Processing (AISP)*, pages 129–135, 2015.

[41] Tim Finin Lisa Mathews Zareen Syed, Ankur Padia and Anupam Joshi. Uco: A unified cybersecurity ontology. *The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence Artificial Intelligence for Cyber Security: Technical Report WS-16-03*, 2016.