# A study on graph neural networks and pretrained models for analyzing cybersecurity texts

By NGUYEN, Chau Minh

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Nguyen Le Minh

September, 2021

# A study on graph neural networks and pretrained models for analyzing cybersecurity texts

By NGUYEN, Chau Minh (1910409)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Nguyen Le Minh

and approved by
Professor Satoshi Tojo

September, 2021 (Submitted)

# Abstract

Analyzing cybersecurity texts is the task of identifying malware actions and determining their characteristics in the documents about cybersecurity threats, utilizing natural language processing (NLP) techniques. This task consists of four subtasks: (1) identifying malware-related sentences (i.e., sentences which describe malware actions) from cybersecurity texts, (2) identifying token labels (i.e., *Malware Action*, *Subject of Action*, *Objectof Action*, and *Modifier of Action*) in malware-related sentences, (3) identifying relation labels (i.e., *Subject-Action*, *Action-Object*, *Action-Modifier*, and *Modifier-Object*) between tokens, and (4) classifying malware actions into attribute labels. The attribute labels are defined and enumerated in the Malware Attribute Enumeration and Characterization (MAEC). Specifically, based on malware's behaviors and attack patterns, MAEC classifies malware actions into four categories, including *ActionName*, *Capability*, *StrategicObjectives* and *TacticalObjectives*; each category includes multiple malware attribute labels: 211 *ActionName* labels, 20 *Capability* labels, 65 *StrategicObjectives* labels, and 148 *TacticalObjectives* labels, results in a total of 444 attribute labels.

Recently, researchers in several disciplines have acknowledged the superior performance of graph neural networks (GNNs). Many NLP researchers also employed GNNs in multiple NLP tasks and achieved promising performance. Besides, pretrained language models are nowadays widely employed because of their robustness in language understanding. In our research, we aim to study on how GNN models and pretrained models can be employed for the task of analyzing cybersecurity texts. Specifically, in this research, we address all four subtasks. The experiment results demonstrate that our proposed models for the subtask 1 and subtask 2 achieve state-of-the-art performance on the *MalwareTextDBv2.0* dataset, which is the largest dataset for malware characteristic analysis.

In the future, we would like to further investigate other methods for subtask 3 and subtask 4. Specifically, in subtask 3, although the rule-based method produces good performance, a supervised model with the help of the self-labeled techniques should benefit the task of identifying relation labels. For the subtask 4, the unannotated data may be exploited to improve the performance of the classifiers.

**Keywords:** cybersecurity texts, analyzing cybersecurity texts, cybersecurity text analysis, graph neural networks, pretrained language models, deep learning

# Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my principal advisor, Professor Nguyen Le Minh, for his constant encouragement, support and kind guidance during my Master course. He has given me a lot of valuable advice, comments, and discussions, which fosters me on my research journey. Without his consistent support, I could not finish the work in this thesis.

I would like to thank committee members consisting of Professor Satoshi Tojo, Associate Professor Kiyoaki Shirai, and Professor Shinobu Hasegawa, for their useful discussions and comments on this thesis.

I would like to thank Associate Professor Kiyoaki Shirai for his valuable advice and suggestions for my minor research project.

I would like to thank Associate Professor Ha Quang Thuy from University of Engineering and Technology, Vietnam National University (Hanoi) for his suggestion and recommendation to study at JAIST.

I would like to express my gratitude that my research is supported in part by the Asian Office of Aerospace RD (AOARD), Air Force Office of Scientific Research (Grant no. FA2386-19-1-4041).

I would like to thank JAIST staff for creating a wonderful environment for both research and life. I would love to devote my sincere thanks and appreciation to all members of Nguyen's laboratory. Being a member of Nguyen's laboratory and JAIST is always a wonderful time of my research life.

I would like to thank the Vietnamese community at JAIST. They have supported me on many aspects of life.

Last but not least, I would like to express my sincere gratitude to my family for supporting me with great patience and love. I would never complete this work without their support.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Cybersecurity Texts Analysis Task

Along with the strong development of the computer era, malware is also released with different malicious purposes. The first types of malware discovered include Creeper (1971), Wabbit (1974), Elk Cloner (first malware on a Mac, released in 1982), Brain (first malware on a computer running MS-DOS, released in 1986). In fact, while computer software is increasingly developed, malware types are also becoming more complex and difficult to detect. With the advent and popularity of the Internet (1983), the spread of malware to personal computers as well as the design of malware to infiltrate the computer systems of organizations was inevitable. There are many examples of malware attacking personal computers and causing serious damage. For example, the ILOVEYOU malware (2000) is said to have been able to reach more than 50 million computers in just 10 days, causing damage estimated in the billions of dollars; or the Blaster malware (2003) has infected hundreds of thousands of computers, causing damage estimated from 2 to 10 billion USD. One of the most dangerous threats to the network of organizations is advanced persistent threat (APT). The sponsors of APTs are typically political organizations, where they aim to attack other political or economical organization via illegeally accessing to victim's computer network, self-covering for not be detected, then performing exploiting or disrupting actions. For example, the existence of Regin (an APT developed by a nation-state actor, which is the National Security Agency of the U.S.) was disclosed in 2014. It is known to attack Belgian telco Belgacom, the German government, and Russian search giant Yandex.

The behaviours as well as the attack methods of malware are documented in many APT reports by cybersecurity companies (e.g., ESET, FireEye, and Kaspersky). Because behaviors and attack patterns of malware are diverse, there should be a specification to distinguish them. Malware Attribute Enumeration and Characterization [17] (MAEC) is introduced for that purpose. It classifies malware into a total of four categories, which are *ActionName*, *Capability*, *StrategicObjectives* and *TacticalObjectives*. Each of the four categories includes many malware attribute labels. Note that a malware may belong to more than one attribute label. The number of attribute labels for each category is

provided as follows:

- Category *ActionName* contains 211 attribute labels.
- Category *Capability* contains 20 attribute labels.
- Category *StrategicObjectives* contains 65 attribute labels.
- Category *TacticalObjectives* contains 148 attribute labels.

The first dataset about malware characteristics analysis using NLP techniques is the *MalwareTextDB* [21] dataset. This dataset is relatively small as only 39 APT reports are chosen to be annotated. After that, a next version of this dataset, *MalwareTextDBv2.0*, was published, which further annotated 46 APT reports (so that the dataset contains 85 annotated APT reports). This dataset also provides hundreds of processed-but-not-annotated APT reports. *MalwareTextDBv2.0* is known as the largest dataset for the task of analyzing malware characteristics. A shared task named SecureNLP [34] (Semantic Extraction from CybersecUrity REports using Natural Language Processing) is held by SemEval to encourage the research on this analyzing task. SecureNLP includes four subtasks as follows:

- **Subtask 1**: Identify malware-related sentences (i.e., sentences which describe malware actions) from APT reports
- **Subtask 2**: Identify token labels (i.e., Malware Action, Subject of Action, Object of Action, and Modifier of Action) in malware-related sentences
- **Subtask 3**: Identify relation labels (i.e., Subject-Action, Action-Object, Action-Modifier, and Modifier-Object) between tokens
- **Subtask 4**: Classify malware actions into attribute labels.

Figure 1.1 shows some examples of how the sentences are annotated for the four subtasks. Specially, in the last example (the sentence is *"an HTTP POST request uploading a segment from edg6EF885E2.tmp"*), the malware action *"uploading"* is determined to belong to all four categories (*Capability*, *ActionName*, *StrategicObjectives* and *TacticalObjectives*). In the annotation process, a malware action can be annotated in more than one categories, however, in each category, this action is classified to only one malware attribute label.

This research aims to analyze cybersecurity texts by addressing the four subtasks proposed by the SecureNLP shared task, on the *MalwareTextDBv2.0* dataset. Specifically, we study on how Graph Neural Network (GNN) models may be employed in the task of cybersecurity texts analysis. Besides, we also employ machine learning models (e.g., Support Vector Machines (SVM)) and pretrained language models (e.g., BERT [5]) to our models.

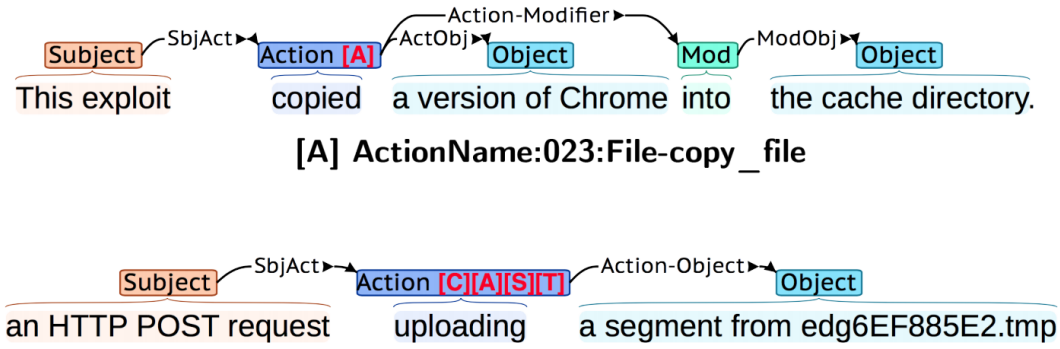


Figure 1.1: Annotation examples for the four subtasks

## 1.2 Graph Neural Networks for Natural Language Processing

In the past decade, deep learning research has developed rapidly and strongly, leading to great success when applied to areas such as computer vision or natural language processing. Deep learning can extract complex patterns from the data thanks to their expressive power. However, initially, studies focused only on extract latent representations for the data in the Euclidean space (e.g., images, texts). Meanwhile, graph-structured data appears and is observable everywhere. For example, the relationships of friends on social networking websites can be represented by social networks, urban road maps can be represented by traffic networks, and the relationships between atoms can also be represented by a graph. However, graph-structured data is non-trivial because they (1) have irregular structures (unlike images or texts), (2) are heterogeneity and diversity (they may contain multiple types of relationships and properties), (3) may be large-scale (eg, e-commerce graphs, social networks), (4) integrated interdisciplinary knowledge (properties of graphs can become very different from graphs in different disciplines: chemistry, biology, or sociology). Over the past five years, the research community has witnessed rapid growth in GNN approaches, ranging from graph recurrent neural networks [46, 30], graph convolutional networks [4, 16, 41], graph autoencoders [37, 39, 47], graph reinforcement learning [8, 20], and graph adversarial methods [42, 7]. Those approaches were developed not only for problems with natural graph-structured data, but also for data such as texts (usually represented by a sequence of tokens) by turning that data into graphs-structured data.

The topic of GNNs for NLP has drawn attention from the research community. Generally, given the texts, the process of tackling an NLP task utilizing GNNs goes through the following steps: (1) graph construction, (2) graph representation learning, and (3) task addressing (Figure 1.2).

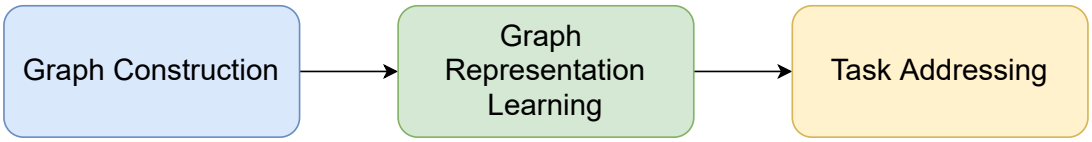


Figure 1.2: GNNs for NLP

## 1.3 Contributions

In this research, we tackle all four subtasks. We achieve state-of-the-art F1 score for the subtask 1 (identifying malware-related sentences) by enriching the features of sentences utilizing external knowledge, with the help of a GNN architecture. For the subtask 2, we implement a BERT-CRF model, which leverages the power of BERT [5] and conditional random field (CRF) [18] to identify token labels. We also exploit the predictions of our model for subtask 1 to help produce the predictions for subtask 2, and we achieve state-of-the-art results for subtask 2. For the subtask 3, we simply apply rules for identifying relation labels. Specially, we construct a graph of the identified tokens and their relations and provide a visualization. For the subtask 4, we propose a model that employs a GNN architecture for tackling the task of classifying malware actions into attribute labels.

## 1.4 Thesis Outline

We organize the structure of this thesis into six parts. The first chapter introduces the thesis. The next four chapters consecutively presents the four subtasks. After that, the last chapter concludes our work. Specifically, the remainder of the thesis is organized as follows:

**Chapter 2** presents our work for subtask 1. We start with the observation of an external knowledge that can help a classifier on classifying malware-related sentences. Next, we propose two methods of leveraging the external knowledge to enrich the features of sentences. After that, we show experimental results and give a conclusion.

**Chapter 3** is to address subtask 2. Firstly, we present the architecture of the BERT-CRF model that we propose to tackle subtask 2. Then we provide experimental results of the model with and without the help from predictions of subtask 1.

**Chapter 4** mainly focuses on how we apply rules for the task of identifying relation labels (subtask 3). We also provide a visualization of the constructed graph from subtask 2 and subtask 3.

**Chapter 5** describes our method for classifying malware actions into attribute labels (subtask 4). Firstly, we present how the graphs are constructed from the texts. Secondly, we present how a GNN model can be employed to tackle the subtask 4. After that, we show the results of the our experiments.

**Chapter 6** shows conclusions and our future works.

# Chapter 2

# Subtask 1: Identify malware-related sentences

## 2.1 Introduction

The subtask 1 is to identify malware-related sentences from the corpus. In other words, given a sentence from an APT report, we need to classify if this sentence is malware-related or not.

In the competition, most of the teams utilized neural network approaches to address subtask 1. While the models following those approaches achieve relatively good performance, a team proposed to utilize a Naive Bayes classifier and can achieve competitive results. This fact indicates that for a challenging task like this subtask, we should focus on choosing features for the sentence representation instead of employing complex neural network models. The previous approaches are as follows:

- Team *Villani* [24] utilized BiLSTM [11] and attention mechanism [25] where they use Glove embeddings [33] to initialize the word embeddings. They got the highest F1 score with F1 score of 57.14%.
- Team *Flytxt_NTNU* [36] ensembled a Naive Bayes classifier with a CRF model. They was the runner-up with the F1 score of 56.87%.
- Team *Digital Operatives* [2] utilized a passive aggressive classifier [3], and achieve F1 score of 51.71%.
- Team *TeamDL* [28] followed [15] to build a convolutional neural network (CNN) on Glove vectors. They achieve F1 score of 50.19%.
- Team *UMBC* [32] utilized a Multi-Layer Perceptron (MLP) model, and achieve F1 score of 17.73%.
- As the outputs of subtask 1 can be generated based on the outputs of subtask 2, the two teams *DM_NLP* [26] and *HCCL* [9] first tackled subtask 2, then produced

Table 2.1: Descriptions and revelant phrases of some attribute labels in *ActionName*

| Category | No. | Name | Description | Relevant Phrases |
|---|---|---|---|---|
| ActionName | 000 | send dns query | Specifies the defined Action of sending a DNS query. | [DNS][query][send] |
| | 001 | send reverse dns lookup | Specifies the defined Action of sending a reverse DNS lookup. | [DNS][reverse lookup][send] |
| | 002 | check for kernel debugger | Specifies the defined Action of checking for the presence of a kernel debugger. | [debug][kernel debugger] |

predictions for subtask 1. They achieved 52.08% and 51.72% on F1 score, respectively.

- There are two teams (*NLP_Foundation* and *NanshanNLP*) who did not provide reports on their methods. As reported in [34], they achieved 49.11% and 15.38% on F1 score, respectively.

The reported results are measured on the *MalwareTextDBv2.0* dataset where the teams solely used the annotated APT reports. However, there is a document in the dataset, named Attribute Reference Guide, which we found containing a short description and a list of relevant phrases for every of 444 attribute labels. We found those information may be helpful for the subtask 1. In this section, we propose to enrich the features of sentence representation with two methods:

- The first method determine the influence of the relevant phrases to the words of a sentence.
- The second method heuristically determine how likely a sentence belongs to each of 444 attribute labels.

The experimental results show that by concatenating our enriched features, we gain more than 9% of F1 score comparing to just using base features, where the classifier is a SVM model.

For this subtask, our contributions are two-fold:

- Proposes to utilize external knowledge for the purpose of enriching features of a sentence, leading to overcomes the limitation of classifying malware-related sentences based solely on annotated APT reports,
- Designs methods for leveraging the knowledge from the Attribute Reference Guide, which is a source of external knowledge, and achieves the highest F1 score on the malware-related sentence classification task.

## 2.2 Related Works

### 2.2.1 Datasets of Annotated Malware Reports

The first dataset about malware characteristics analysis using NLP techniques is the *MalwareTextDB* [21] dataset. This dataset is relatively small as only 39 APT reports are chosen to be annotated. After that, a next version of this dataset, *MalwareTextDBv2.0*, was published, which further annotated 46 APT reports (so that the dataset contains 85 annotated APT reports). This dataset also provides hundreds of processed-but-not-annotated APT reports. *MalwareTextDBv2.0* is known as the largest dataset for the task of analyzing malware characteristics.

### 2.2.2 Graph Attention Networks

Graph Neural Networks (GNNs) have received great attention from the research community in recent years. Many GNN architectures (e.g., Graph Convolutional Networks [16], GraphSAGE [12], and GAT [40]) were proposed for several tasks (e.g., node classification, link prediction) on graph-structured datasets, in both transductive and inductive settings. GATs (Graph ATtention networks) stand out of those architectures with an impressive performance on an inductive node classification task on a PPI (protein-protein interaction) dataset. It achieved the highest accuracy of 97.3% on this dataset while previous approaches achieved 76.8% or less. This fact demonstrates that GATs can effectively learning node representations.

GATs is constructed by stacking masked self-attentional layers. The masked self-attentional layers allow each node to implicitly determine weights to the nodes in its neighborhood via attending to its neighborhoods' features. Besides, the multi-head attention is employed for learning self-attention. The features of a node is transformed after feeding through a layer as:

$$\vec{h}_i^{'} = \Big\Vert_{k=1}^{K} \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \boldsymbol{W}^k \vec{h}_j \right) \tag{2.1}$$

Specially, as the outputs of the final layer are used for classification, GATs employ averaging instead of concatenation for the final layer as follows:

$$\vec{h}_i^{'} = \sigma \left( \frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \boldsymbol{W}^k \vec{h}_j \right) \tag{2.2}$$

where $\Vert$ denotes concatenation, $K$ is the number of attention heads, $\sigma$ is a non-linearity, $\mathcal{N}_i$ is the neighborhood of node $i$, $\alpha_{ij}^k$ is a normalized attention coefficient calculated by the $k$-th attention head, $\boldsymbol{W}^k$ is the corresponding input's weight matrix.

## 2.3 Methods

In this section, we propose two methods that leverage the knowledge (specifically, the lists of relevant phrases) from the Attribute Reference Guide to enrich the features of a sentence.

- The first method is based on the assumption that the likelihood of a sentence to be a malware-related sentence should be higher if the words in the sentence pay more attention to the set of relevant phrases taken from the list of relevant phrases (in this set, all phrases are unique). Therefore, after acquiring the relevant phrase set, a GAT model is employed on a graph which is constructed based on the words in the sentence and the phrases from the set of relevant phrases, for the purpose of producing a *weak label* for the sentence. We use this *weak label* as an enriched feature.

- The second method is based on the assumption that if a sentence contains high percentage of relevant phrases in the list of relevant phrases of an attribute label, then this sentence has a high chance to belong to this attribute label. To this end, we design a function to measure how likely a sentence should belongs to an attribute label. There are a total of 444 enriched features corresponding to 444 attribute labels.

A high-level overview of our approach is shown in the Figure 2.1.

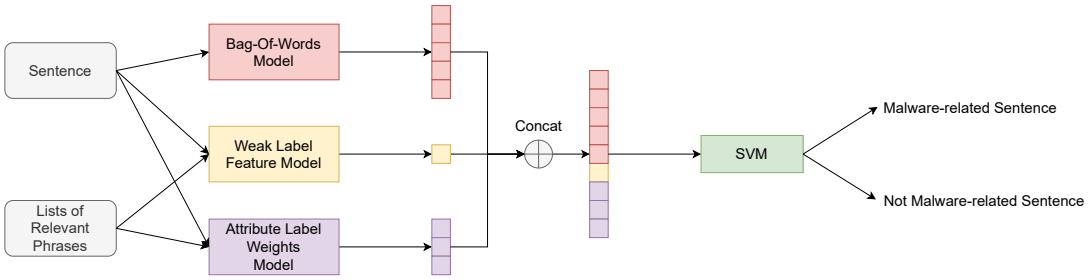


Figure 2.1: A high-level overview of our approach

### 2.3.1 Data Preprocessing

We perform a data preprocess phrase to all the sentences in the data as follows:

- non-alphanumeric removal
- character lowercased
- stopwords removal

### 2.3.2 The Classifier and the Base Features

As aforementioned, choosing features for representing a sentence should be focused, not employing complex classifiers. For that reason, we utilize support-vector machine (SVM), a traditional machine learning (ML) model, as the classifier. Regarding the kernel of this model, we choose the Radial Basis Function (RBF) kernel. For the base features, we use BOW features. After producing the enriched features, we will concatenate them to the base features. Regarding the hyperparameters used for the SVM classifier, we perform two-step grid search, then choose based on the model's performance on the development set.

### 2.3.3 Method 1: Determining Weak Labels for Sentences

Each malware attribute label in the Attribute Reference Guide has several relevant phrases. In fact, many phrases exists in more than one relevant phrase list (each list corresponds to an attribute label). After remove duplicating phrases, we obtain 444 unique relevant phrases. Then, for a sentence, a graph is constructed which contains word nodes (i.e., the words from the sentence), and phrase nodes (i.e., the phrases in the obtained relevant phrase set). The constructed graph is undirected. Regarding constructing edges, we construct as follows:

- Link word nodes with phrase nodes,
- Self-loop for all nodes (in such a way, a node can attend to itself, so that the node may retain its own information for the next layer of GATs),
- Link all word nodes together (to that the graph contains intra-sentence information).

We provide an example in Figure 2.2 with the sentence *"They work together to gain access to their targets and steal data"*, showing how the edges are constructed.

The constructed graph will be fed through a GAT model for the task of node classification. In here, we describe how we design node labels. As a way to identify malware-related sentences, if a sentence describes a malware action or more, then this sentence is classified as a malware-related sentence. Unfortunately, we cannot extract the tags for malware actions based on the annotated data as it does not provide such information. The annotated data only provides the label of the whole sentence, but not for the malware action in this sentence. Hence, the tags for malware actions are heuristically created as follows:

- For a malware-related sentence, we use NKTK[1] to obtain Part-Of-Speech (POS) tags of words in the sentence. After that, we assign positive label to all the word nodes corresponding to the words that have POS tag beginning by "VB" (i.e., indicating a verb), while those with other kind of tags are assigned with negative label.
- For a non-malware-related sentence, we also get POS tags, then assign negative label to all word nodes.

[1]https://www.nltk.org/

After the graph is constructed, we employ a GAT model to implicitly learn the node representation on the node classification task. The trained GAT model shall be used to predict nodel labels in the testing phase. We use the predictions to obtain the *weak label* of a sentence as follows:

- If one or more positive label is predicted for word nodes, we assign the value 1 for the *weak label*.
- Otherwise, we assign the value 0 for the *weak label*.

Then, we concatenate the *weak label* to the base features (BOW features) to help addressing the malware-related sentence classification task.

### 2.3.4 Method 2: Generating Attribute Label Weights for Sentences

As mentioned above, if a sentence describes a malware action or more, then this sentence is classified as a malware-related sentence. In our task, the malware action should be determined to be in at least one malware attribute labels (among 444 attribute labels). As shown in the Attribute Reference Guide, there are a short description about malware malware behavior and a list of relevant phrases, for each attribute label. In this method, we assume that the sentence is likely to be classified into an attribute label if this sentence contains all or almost relevant phrases of an attribute label. Based on this assumption, we design a formula to calculate a score $s_i$ $(1 \leq i \leq 444)$, demonstrating how likely a sentence is classified to the attribute label $i$. This formula is designed for a sentence $S$ as follows:

$$s_i = \frac{1}{\|P_i\|} \sum_{p_j \in P_i} f(p_j, S) \tag{2.3}$$

where $P_i$ indicates attribute label $i$'s list of relevant phrases, $\|P_i\|$ specifies the number of phrases in the list $P_i$, and $f(p, S)$ is defined as a function that returns 0 if the phrase $p$ is not in the sentence $S$ and returns 1 otherwise.

The scores $s_i$ $(1 \leq i \leq 444)$ is then concatenated to the base features and the *weak label* feature, so that the representation of a sentence is enriched by two methods.

### 2.3.5 Training and Predicting

After concatenation of the enriched features from the two methods for the sentences in the dataset, we use a SVM model to train. For the hyperparameters $n_{vocab}$, $C$, $\gamma$, we perform a two-step hyperparameter search, then pick the values of hyperparameters based on the model performance on the developement set. After that, we use the SVM model with the picked hyperparameters to test on the test set.

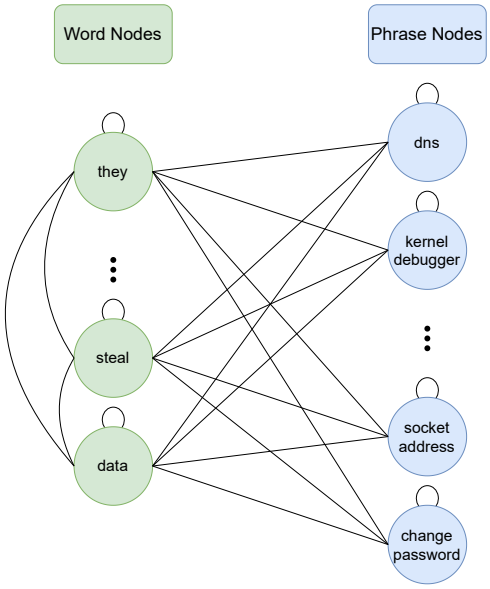


Figure 2.2: Example of how the edges are constructed

## 2.4 Experimental Results

### 2.4.1 Data

We use the data split for training, development, and test data of organizer of the SecureNLP competition. Data statistics on the data is shown in Table 2.2.

### 2.4.2 GAT Hyperparameters

As suggested in [10], we set the GAT model with 3 layers, where the number of heads for the layers are 4, 4, 6, respectively. The number of hidden features is set to 256. We train the model for 200 epochs with early stopping.

### 2.4.3 Hyperparameter Choosing for SVM

After the preprocessing phase, we obtain $12,975$ words from $9,424$ training sentences. In here, we present a two-step hyperparameter search to find an appropriate setting for $n_{vocab}$ (number of words in the BOW vocabulary), $C$ (regularization parameter of the SVM model), and $\gamma$ (kernelcoefficient for RBF). (We also conducted experiments with SVM model with linear kernel, but the average performance of the same settings are consistently lower than those with RBF kernel, so in this section, we only present about SVM with RBF kernal.) Since the range of searching for the value of $n_{vocab}$, $C$ and $\gamma$ is wide, we employ the two-step grid search as follows:

- The first grid search shall search on a wide range of values, to find a smaller searching range, where the performance on the developement set is stable and high comparing

to all other ranges.

- The second grid search shall perform grid search in the specified smaller range to find the most appropriate setting, choosing based on tuning on the development set. This setting shall be applied to the test data.

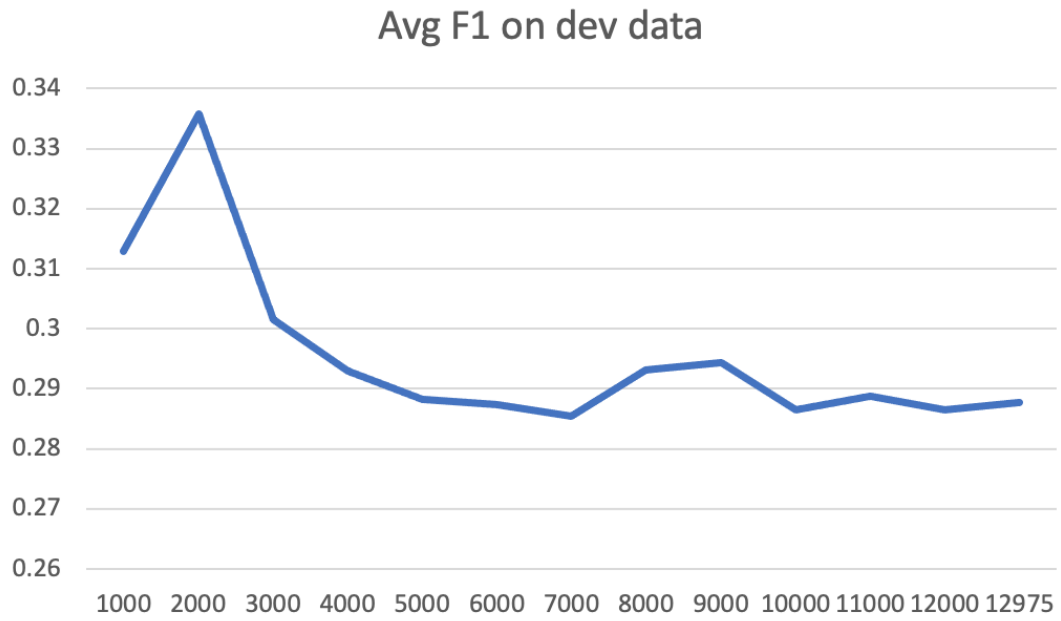


Figure 2.3: First hyperparameter search: Average F1 score on the development data

In the first grid search, we perform searching with the following hyperparameters:

- $n_{vocab} \in \{1000, 2000, 3000, ..., 11000, 12000, 12975\}$
- $C \in \{1e{-}3, 1e{-}2, 1e{-}1, 1, 5, 10, 50, 100, 1000\}$
- $\gamma \in \{1e{-}5, 1e{-}4, 1e{-}3, 1e{-}2, 1e{-}1, 1, 10, 100\}$

Figure 2.3 demonstrates the average of the performance of the SVM model on the development set. We can see that, when $n_{vocab} = 2000$, this average value is highest. In case $n_{vocab} = 2000$, we found the settings of $C$ in range $[1, 5]$, and $\gamma$ in range $[1e{-}5, 1e{-}4]$ would be most appropriate to further search. Thus, in the second grid search, we set $n_{vocab} = 2000$, and further search with:

- $C \in \{1, 2, 3, 4, 5\}$
- $\gamma \in \{1e{-}5, 3e{-}5, 5e{-}5, 7e{-}5, 9e{-}5\}$

Table 2.2: Data statistics

| **Data** | **# sentences** | **# non-malware -related sentences** | **# malware-related sentences** | **% malware-related sentences** |
|---|---|---|---|---|
| Training | 9424 | 7220 | 2204 | 23.39 |
| Development | 1213 | 1134 | 79 | 6.51 |
| Test | 618 | 528 | 90 | 14.56 |

Table 2.3: Experimental results for subtask 1

| **Model** | **Acc** | **Prec** | **Recall** | **F1** |
|---|---|---|---|---|
| Villani [24] | 54.47 | 47.76 | 71.11 | 57.14 |
| Flytxt_NTNU [36] | 85.28 | 49.59 | 66.67 | 56.87 |
| DM_NLP [26] | 79.45 | 39.43 | **76.67** | 52.08 |
| HCCL [9] | 86.41 | 53.57 | 50.00 | 51.72 |
| Digital Operatives [2] | 79.45 | 39.31 | 75.56 | 51.71 |
| TeamDL [28] | 79.13 | 38.46 | 72.22 | 50.19 |
| NLP_Foundation | 76.86 | 36.13 | **76.67** | 49.11 |
| UMBC [32] | 41.42 | 11.14 | 43.33 | 17.73 |
| NanshanNLP | 71.52 | 13.56 | 17.78 | 15.38 |
| SVM on BOW + Enriched Features | **87.22** | **55.56** | 61.11 | **58.20** |

Table 2.4: Ablation Experiments

| **Features** | **Acc** | **Prec** | **Recall** | **F1** |
|---|---|---|---|---|
| BOW | 87.54 | 60.66 | 41.11 | 49.01 |
| BOW + Weak Label Feature | 87.54 | 58.23 | 51.11 | 54.44 |
| BOW + Attribute Label Weights | 86.89 | 54.55 | 60.00 | 57.14 |

After performing two extensive grid searches, we found the setting with $n_{vocab} = 2000$, $C = 1$, and $\gamma = 9\text{e}{-5}$ to be the most appropriate in our case, where it reached the highest F1 score on the development set. This setting is used for the test set. As we can see in Table 2.3, the SVM model on our enriched features achieves the state-of-the-art (SoTA) F1 score, surpasses the previous SoTA achieved by team *Villani* by 1.06%.

For investigating the contribution of enriched features generated by the two proposed methods, ablation experiments are performed (Table 2.4). We can see that when we use only the base features (BOW features), the F1 score is only 49.01%. However, with the concatenation of the *weak label* feature, we gain 5.43% of F1 score. This value is 8.13% if we concatenate the attribute label weights instead of the *weak label* feature. Especially, if we concatenate both types of enriched features, the improvement is 9.19%, where our F1 score achieves 58.20%.

## 2.5 Conclusion

In this chapter, two methods are presented for the purpose of exploiting knowledge in an external document (the Attribute Reference Guide) to produce enriched features for sentences. One method employs GATs to obtain a *weak label* for a sentence, where the other produce 444 weights corresponding to 444 attribute labels, to further help representing the sentences. With those features, we gain an improvement of about 9% comparing to the performance of our SVM model without those features.

# Chapter 3

# Subtask 2: Identify token labels

## 3.1 Introduction

In subtask2, special tokens in a relevant sentence had to be identified and labeled with one of the following token labels (refer to Figure 3.1):

- Action: This refers to an event, such as "implements", "deploys", and transferred".
- Entity: This refers to the initiator of the Action such as "Babar" and "they" or the recipient of the Action such as "an obfuscation technique", "privilege-escalation tools", and "the data"; it also refers to word phrases that provide elaboration on the Action such as "hide certain API names" and "external FTP servers".
- Modifier: This refers to tokens that link to other word phrases that provide elaboration on the Action such as "to".

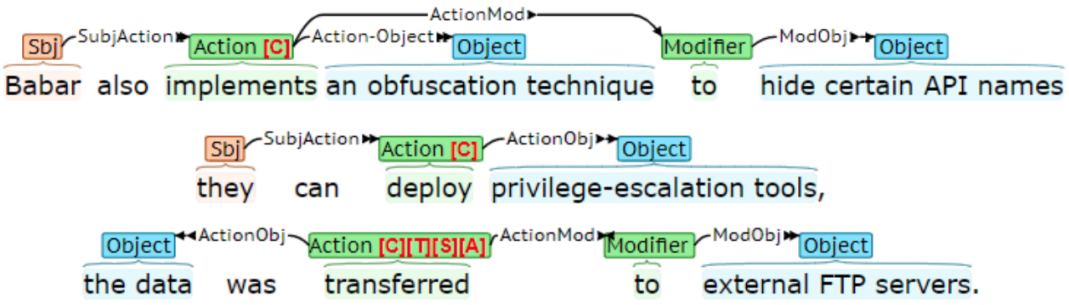


Figure 3.1: Annotation examples for the four subtasks

In the competition, many team submitted their outputs for subtask 2. Team *Flytxt_NTNU* [36] utilized CRF [18] to address this subtask, following a two-step method: firstly, they determine if a token is a *mention token* (i.e., a token is either *Action*, *Entity*, or *Modifier*), then they classify those tokens into their specific token labels. Team *DM_NLP* [26] proposed a hybrid approach, where they employed BiLSTM-CNN-CRF, following by

the Bi-directional LSTM-CNNs-CRF model in [27]. They made use of multiple other features, including part-of-speech (POS), dependency labels, chunk labels, NER (Named Entity Recognition) labels, and brown clustering labels. Team *HCCL* [9] tackled subtask 2 by employing the BiLSTM-CNN-CRF model with the use of POS featuers. Team *Digital Operatives* [2] proposed a linear CRF approach that leverages features from POS, lemma, dependency links, and bigrams. Team *TeamDL* [28] utilized a CRF model with multiple features, such as POS tags, N-grams, word lemmas, and word shape features. Team *UMBC* [32] also employed CRF model with multiple features, quite similar but less than those used by team *TeamDL*. Regarding the results, Team *DM_NLP* achieved the best results on both metrics with F1 normal score is 29.23% and F1 relaxed score is 39.18%.

It is understandable that the CRF approach are popular in the subtask 2 because of its robustness for the task of sequence labeling. We also utilize the CRF approach for our method, however, we implement it in an neural network manner as the deep learning approach has demonstrated its outstanding performance on a large range of pattern learning tasks. However, the limited data should be one major obstacle if we only follow the above approach. So, we employ a BERT model to our method, so that we can leverage its ability of language understanding. Specifically, we implement a BERT-CRF model for the subtask 2.

In this chapter, we tackle subtask 2 by using BERT-CRF model with the support of subtask 1's result (which is produced a binary classification model). The results show that this approach achieves the best results on both normal scores and relaxed scores on subtask 2.

## 3.2 Related Works

### 3.2.1 Some Transformer-based Pretrained Models

Pretrained models have been widely used because of their robustness and effectiveness. BERT can be considered one of the breakthrough in NLP when it was first released. After that, many new models have been proposed to make BERT more robust and/or effective. RoBERTa [31] proposes dynamic masking and removes the BERT's Next Sentence Prediction (NSP) task. They state that BERT is under-pretrained, and so they pretrained on larger corpus for longer time, and make the pretrained models more robust comparing to BERT models. AlBERT [19] proposed two techniques to not only lower the comsumption of memory, but also increate the speed of training for the BERT model. It is known as a *lite* version of BERT. DistilBERT [43] used a distillation technique, which reduced the size of the model, while retaining almost its capability of language understanding and being faster.

### 3.2.2 CRF++

CRF++[1] is a simple, customizable, and open source implementation of Conditional Random Fields (CRFs) for segmenting/ labeling sequential data. It is designed for generic purpose and will be applied to a variety of NLP tasks, such as Named Entity Recognition, Information Extraction and Text Chunking.

### 3.2.3 HuggingFace's Transformers

HuggingFace's Transformers[2] provides state-of-the-art general-purpose architectures for Natural Language Understanding and Natural Language Generation with over thousands of pretrained models. We used many of those models for subtask 2.

### 3.2.4 GCDT

GCDT (Global Context enhanced Deep Transition architecture) [23] if proposed for sequence labeling. GCDT deepens the state transition path at each position in a sentence, and further assign every token with a global representation learned from the entire sentence.

## 3.3 Method

While many teams in the competition chose to tackle subtask 1 and 2 separately, we propose to train a binary classification model for Subtask 1 and BERT-CRF model for the preliminary results of Subtask 2, after that, use the Subtask 1's result to support producing the final result of Subtask 2. Specifically, we will label all the tokens of the sentences which are predicted as label 0 in subtask 1 as label 'O'. Before training on the BERT-CRF model, we preprocess the data so that all tokens are labelled in BIOES format instead of BIO format. Figure 3.2 shows an overview of our approach for subtask 2.

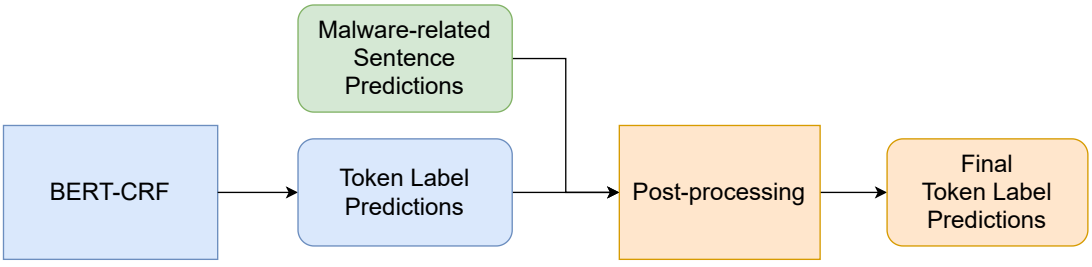


Figure 3.2: Our approach for subtask 2

[1]https://taku910.github.io/crfpp/
[2]https://github.com/huggingface/transformers

We implement a BERT-CRF model to tackle with the sequence labelling task. The BERT-CRF model is implemented based on the BERT model of HuggingFace's Transformers and the CRF layer by AllenNLP[3]. Figure 3.3 show the overview of the BERT-CRF model.

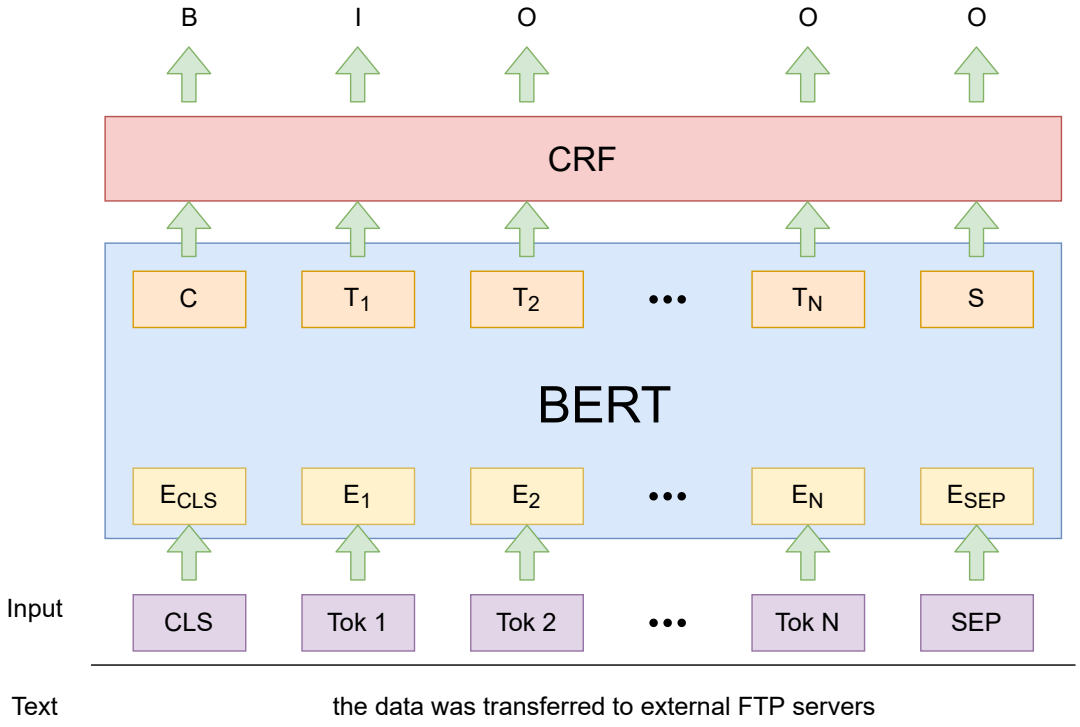


Figure 3.3: An overview of the BERT-CRF model

## 3.4 Experimental Results and Analysis

### 3.4.1 Experimental Results

We use *bert-base-based* as the pretrained model of BERT. The BERT-CRF model then is finetuned on the data for subtask 2, and then is used for predicting token labels. Next, we use predictions of subtask 1 as a post-processing phase. As mentioned above, all the tokens of the sentences which are predicted as label 0 in Subtask 1 will be labelled as 'O'. Table 3.1 shows the experimental results of Subtask 2. The experimental results show that the performance of our BERT-CRF model is competitive with the performance of the hybrid model proposed by team *DM_NLP*. However, with the help of predictions of our model for subtask 1, we achieve better performance on both normal F1 score and relaxed F1 score. Specifically, we improve 0.91% on normal F1 score, and up to 3.70% on relaxed F1 score comparing the the older state-of-the-art on subtask 2.

[3]https://github.com/allenai/allennlp

Table 3.1: Experimental results of subtask 2

| Method | F1 (Normal Scores) | F1 (Relaxed Scores) |
|---|---|---|
| DM_NLP | 29.23 | 39.18 |
| Flytxt_NTNU | 27.56 | 36.17 |
| NLP Foundation | 27.52 | 38.62 |
| TeamDL | 25.3 | 35.81 |
| UMBC | 22.2 | 31.98 |
| HCCL | 21.72 | 37.6 |
| NanshanNLP | 21.18 | 28.03 |
| Digital Operatives | 15.51 | 24.96 |
| BERT-CRF (Ours) | 28.75 | 39.04 |
| BERT-CRF with post-processing (Ours) | **30.14** | **42.88** |

To further compare our BERT-CRF model with other models for sequence labelling, we also report the results of multiple Transformer-based models (BERT, RoBERTa, AlBERT, DistilledBERT), as well as CRF++ and GCDT. Table 3.2 shows the experimental results. We can see that, except GCDT, other models (many Transformer-based pretrained models and the CRF++ model) perform with not-so-competitive performances. It is noteworthy to mention that GCDT achieves better results comparing to team *DM_NLP*, however, with a small margin.

In all experiments which involve a Transformer component, we set the max sequence length to be 128.

### 3.4.2 Analysis

Although our approach improves the performance comparing to previous approaches, the results are still relatively low. We investigate on the causes of low experimental results. Subtask 2 requires the participants to label tokens as "Entity", "Action" or "Modifier". However, only the sentences that are classified as the relevant sentences for inferring malware actions and capabilities should be labeled with the token label rather than 'O'. Even in those sentences, just a few tokens should be labeled with the token label rather than 'O'. Table 8 shows an example of gold label for the sentence *"The first binary payload that lands on the system is relatively simple and serves as a method of yet again detecting the operating system version and [. . . ]"*. Furthermore, in some cases, the gold labels are inconsistent. Figure 3.4, Figure 3.5 and Figure 3.6 give some examples.

## 3.5 Conclusion

The BERT-CRF model achieves the highest on subtask 2 with the support of the binary classification model (from subtask 1). However, the results are still relatively low because the models can not learn the latent restrictions on subtask 2.

Table 3.2: Experimental results of other models for sequence labelling task

| Method | F1 (Normal Scores) | F1 (Relaxed Scores) |
|---|---|---|
| BERT (bert-base-cased) | 25.51 | 33.75 |
| RoBERTa (roberta-base) | 27.70 | 34.03 |
| AlBERT (albert-base-v1) | 26.48 | 34.11 |
| AlBERT (albert-base-v2) | 27.61 | 35.23 |
| DistilBERT (distilbert-base-cased) | 24.34 | 32.75 |
| CRF++ with POS tags | 23.16 | 36.26 |
| GCDT with bert-base-cased embeddings | 29.95 | 39.42 |

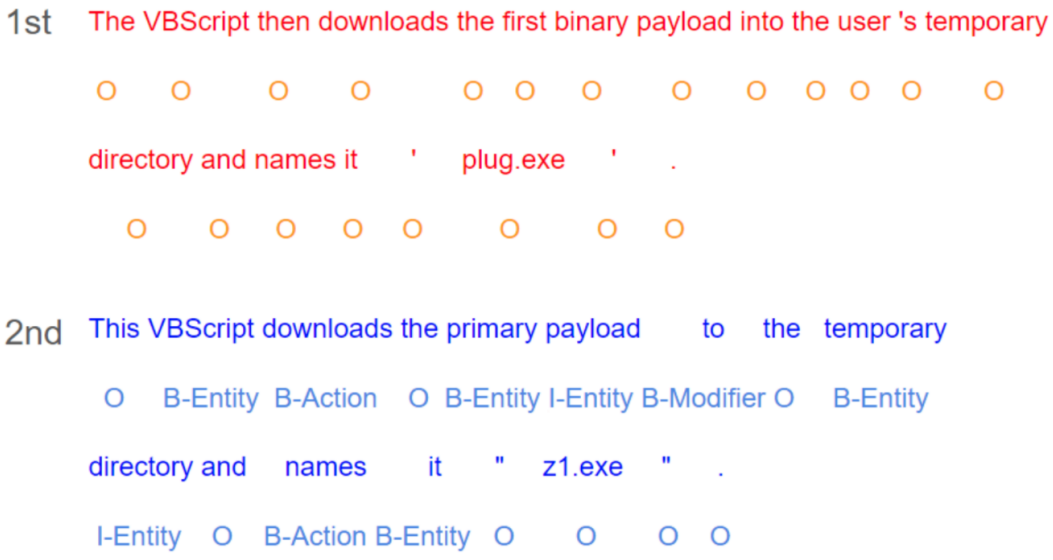


Figure 3.4: An example of two nearly identical sentences are annotated very differently

| Token | Label | Token | Label |
|---|---|---|---|
| They | B-Entity | upload | B-Action |
| have | O | files | B-Entity |
| the | O | , | O |
| ability | O | download | B-Action |
| to | O | files | B-Entity |
| retrieve | B-Action | , | O |
| detailed | O | and | O |
| bot | B-Entity | self- | B-Action |
| / | I-Entity | delete | I-Action |
| system | I-Entity | the | O |
| information | I-Entity | malware | B-Entity |
| , | O | from | B-Modifier |
| update | B-Action | the | O |
| bot | O | system | B-Entity |
| configuration | B-Entity | . | O |
| , | O | | |

Figure 3.5: An example of the case when entity's descriptive word does not count toward the whole entity

| Token | Label |
|---|---|
| The | O |
| first | B-Entity |
| binary | I-Entity |
| payload | I-Entity |
| that | O |
| lands | O |
| on | O |
| the | O |
| system | O |
| is | O |
| relatively | O |
| simple | O |
| [...] | [...] |

Figure 3.6: An example of the case when the word "the" does not count toward the whole entity

# Chapter 4

# Subtask 3: Identify relation labels

## 4.1 Introduction

In subtask 3, participants were asked to identify the relation between the tokens. The gold labels for the tokens are provided here due to the low performance of the initial models on subtask 2. The relation labels are described as follows:

- *SubjAction* links an *Action* with its relevant *Subject*.
- *ActionObj* links an *Action* with its relevant *Object*.
- *ActionMod* links an *Action* with its relevant *Modifier*.
- *ModObj* links a *Modifier* with the Object that provides elaboration.

Figure 4.1 shows an example.

In the competition, no team participated in the subtask 3.

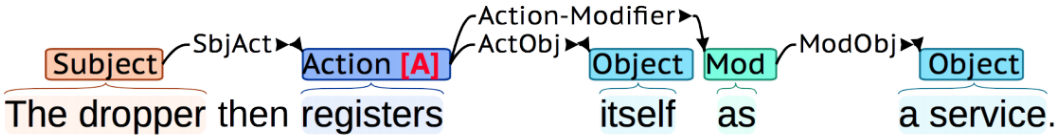


Figure 4.1: An annotation example

## 4.2 Visualization

In this section, we provide an visualization of the graph created by the tokens and their relations. To construct a graph, we need nodes and edges. In this case, nodes are *Entities*, *Actions* and *Modifiers*, and edges are the relations between them (one of the four relations: *SubjAction*, *ActionObj*, *ActionMod*, *ModObj*).

Logger module is a component of the PVZ (PVZ is shorthand for Parviz, one of the members of the Cleaver team) bot tool chain. When executed, it will capture the user's keystrokes and save them to a location which PVZ bot then exfiltrates.

Figure 4.2: An example about coreference issue

During the visualization phase, we face a coreference issue. Figure 4.2 gives an example where the phrases in a same color refer to the same object, and they should be the same node in the constructed graph. We utilize HuggingFace's NeuralCoref 4.0[1]) to address this coreference issue. After that, each *Entity*, *Action* and *Modifier* will be a node in the graph. To create edges, we apply rule-based method following [21] as follows:

- If a *Modifier* is followed by an *Entity*, a *ModObj* relation is predicted between the *Modifier* and the *Entity*.
- If an *Action* is followed by an *Entity*, an *ActionObj* relation is predicted between the *Action* and the *Entity*.
- If an *Entity* is followed by an *Action* of token length 1, a *SubjAction* relation is predicted between the *Entity* and the *Action*.
- An *ActionObj* relation is predicted between any *Action* that begins with be and the most recent previous *Entity*.
- An *ActionObj* relation is predicted between any *Action* that begins with *is*, *was*, *are* or *were* and ends with *-ing* and the most recent previous *Entity*.
- An *ActionMod* relation is predicted between all *Modifiers* and the most recent previous *Action*.

This rule-based method was used as the baseline for subtask 3 [34]. The reported results of the method is shown in Table 4.1. As we can see, with simple rules, the F1 score achieves the number of 85.71%. Finally, we implement the visualization with *pyvis*[2].

Figure 4.3 shows the overview of the visualized graph. As we can see, in the center of the visualization, there are some subgraphs with much more links than the subgraphs near the margin. It is because, in many cases, an *Entity* of one sentence can be the *"Object"* of of another sentence. Hence, the links continue to get longer. Figure 4.4 show an example of the differences.

[1]https://github.com/huggingface/neuralcoref
[2]https://pyvis.readthedocs.io/

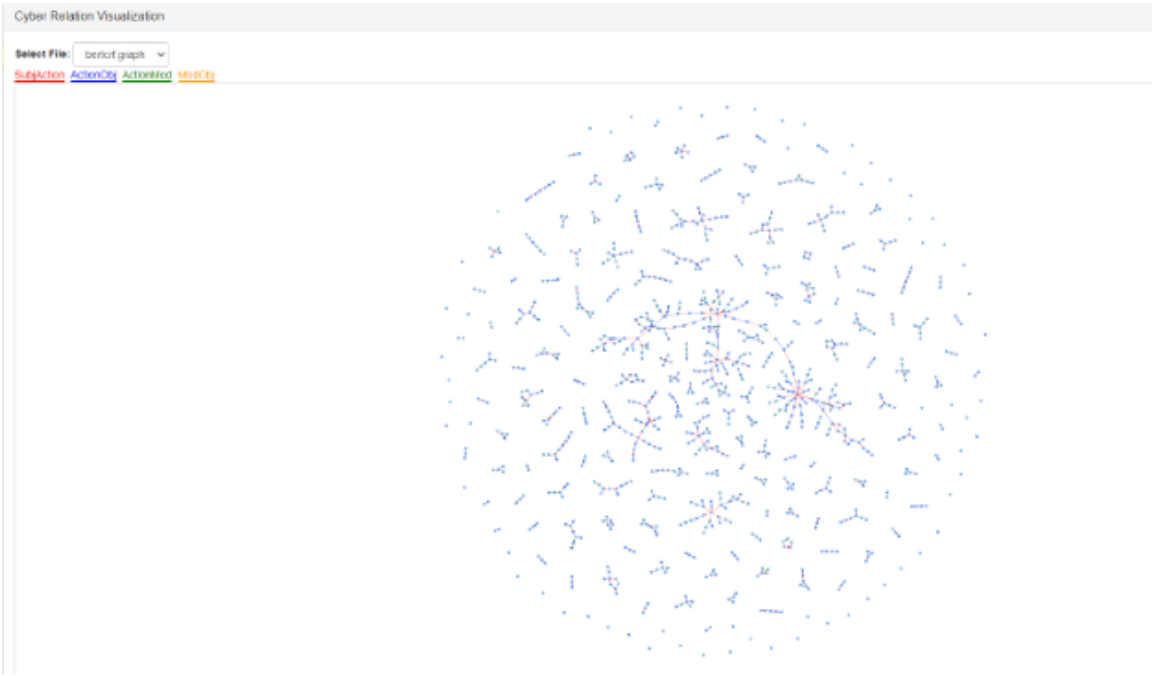


Figure 4.3: The overview of the graph visualization

Table 4.1: Results of the rule-based method

| Method | Prec | Recall | F1 |
|---|---|---|---|
| Rule-based baseline | 85.60 | 85.83 | 85.71 |

## 4.3 Conclusion

In this section, we utilize the rule-based method [21] to create relation labels. After that, we address the coreference issue to construct a graph and visualize it.

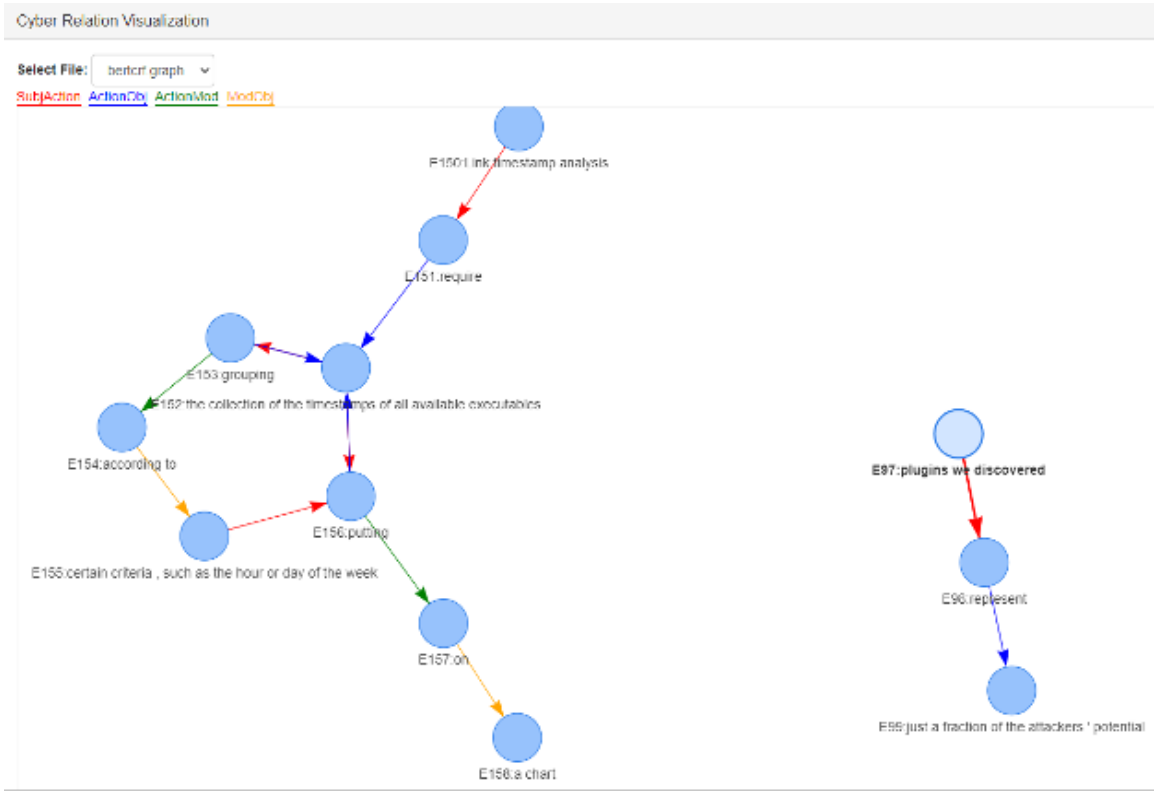


Figure 4.4: Example of subgraphs with multiple edges or with a few edges

# Chapter 5

# Subtask 4: Classify malware actions into attribute labels

## 5.1 Introduction

For the malware action classification task, Word Annotation Embedding (WAE) [35], inspired by the idea of word embedding [29], was introduced with the use of attribute-label keywords in a semi-supervised learning paradigm to learn embeddings of words and attribute labels. The learned embeddings were then used to feed into classifiers for classification. Attribute-label keywords of an attribute label are the words associated with this attribute label, determined via the MAEC specification and the annotated data.

Recently, researchers in several disciplines have acknowledged the superior performance of graph neural networks (GNNs). Many NLP researchers also employed GNNs in multiple NLP tasks, including text classification [45, 44, 22, 6], and achieved desirable results. Among those models, HyperGAT [6] demonstrated its superiority in both text classification performance and computational efficiency by employing the idea of hyperedge (hyperedge can link 2 nodes or more) and dual attention mechanism (based on attention mechanism[25]). In [6], each document is represented by a hypergraph where the nodes are the words and the hyperedges are constructed based on the relations between the words. There are two types of hyperedges: sequential hyperedge and semantic hyperedge. A sequential hyperedge connects consecutive words in a document. Specifically, in [6], all the words of a sentence in a document is connected by a sequential hyperedge. Semantic hyperedges are used to enrich the semantic context of the words. Specifically, a semantic hyperedge connects words in a document if the words are topic-related (determined based on latent topic mined via LDA [1]).

HyperGAT demonstrated its superior text classification performance on many datasets; however, for a challenging task (e.g., malware action classification) on a low-resource dataset (e.g., MalwareTextDB dataset), its performance may be limited. In that case, the use of domain feature may help HyperGAT improve the classification performance. Attribute-label keywords are demonstrated in [35] to carry essential information for classifying malware actions. Therefore, hyperedges constructed based on attribute-label key-

words (called malware-attribute hyperedges) may be useful for HyperGAT on this task. In this research, we (i) investigate the contribution of malware-attribute hyperedges, and (ii) validate the superiority of sequential hyperedge to sequential normal-edge (edge that connects two nodes) in HyperGAT for the task of malware action classification. Malware-attribute hyperedges are hyperedges constructed based on attribute keywords. The details on hyperedge construction are described in Section 5.3. The experimental results demonstrates that malware-attribute hyperedges consistently help improve classification performance of HyperGAT, and hyperedge can embed more information than normal edge.

The remainder of the chapter is organized as follows. Section 5.2 walks through related work. Section 5.3 describes the methods. Section 5.4 presents the experimental results and analysis. Finally, Section 5.5 concludes our work.

## 5.2 Related Works

MalwareTextDB dataset [21] is the first attempt to annotating malware-related texts for analyzing malware characteristics using NLP techniques. As a baseline, a bag-of-words model was used to represent token groups. After that, two machine learning models (i.e., linear support vector machine and multinomial Naive Bayes) were used for malware action classification.

Roy et at. [35] introduced a system that learns the embedding of both words in APT reports and attribute labels in the MAEC specification. First, based on the MalwareTextDB dataset and the MAEC specification, each word in the dataset is heuristically linked to several attribute labels. After that, based on the idea of word embedding, Word Annotation Embedding (WAE) [35] is proposed for learning the embeddings. Specifically, they link a word with its context words, but, in addition, they also link this word with its attribute labels specified in the first step. They proposed employing label-aware negative sampling (based on the idea of negative sampling) for learning the WAE model. Finally, the learned embeddings were used as the input for SVM (Support Vector Machine)[13] for classification task.

Although WAE can make use of attribute labels for the embedding learning process, it shows no attempt to leverage long-distance relationship between words. In other words, only the words inside a context windows size are considered during learning. However, in case long-distance relationship between words contribute important information regarding attribute label of the malware action, WAE will miss this feature. The long-distance relationship issue can be addressed by using hyperedges, when we construct appropriate hyperedges. HyperGAT framework allows improving node connection quality via adding more hyperedges. It is noted that malware action classification can be a challenging task due to the huge number of malware attribute and the low-resource dataset. Hence, adding hyperedges that contain essential task-related information should have positive impact on a task with limited resource.

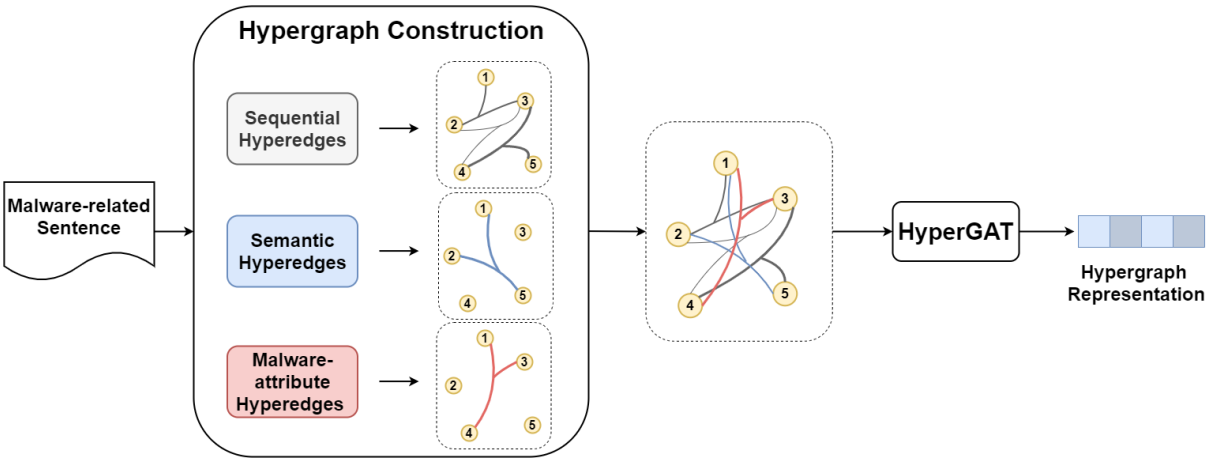


Figure 5.1: High-level view of the framework. Different hyperedges marked by different color or line size.

## 5.3 Methods

In this section, we present about (1) investigating the contribution of malware-attribute hyperedges in HyperGAT for the task of malware action classification, and (2) validating the superiority of sequential hyperedge to sequential normal-edge.

### 5.3.1 HyperGAT for malware action classification

In this subsection, we present about constructing hypergraphs from malware-related sentences, and leveraging HyperGAT network [6] for learning representations of hypergraphs. Figure 5.1 illustrates a high-level view of the framework. The details are described as follows.

**Hypergraph construction.** Hypergraph is a special type of graph, in which its hyperedges can connect more than 2 nodes. For each malware-related sentence, a hypergraph is constructed where the nodes are the words from the sentence and the hyperedges consists of sequential hyperedges, semantic hyperedges, and malware-attribute hyperedges. To construct sequential hyperedges, we slide a fixed-size sliding window over a malware-related sentence, and connect words inside the window by a hyperedge each time. Semantic hyperedges are used to capture the semantic context in a document. We follow [6] for constructing semantic hyperedges. Specifically, we use LDA [1] to mine latent topics in a document, then takes the top $N$ words of each latent topic to represent that latent topic. Next, words in the document that belong to any latent topic (i.e., in the list of words that represents a latent topic) are connected by a hyperedge.

**Hypergraph representation learning.** After hypergraph construction, the hypergraph is fed into HyperGAT for learning representation and classification.

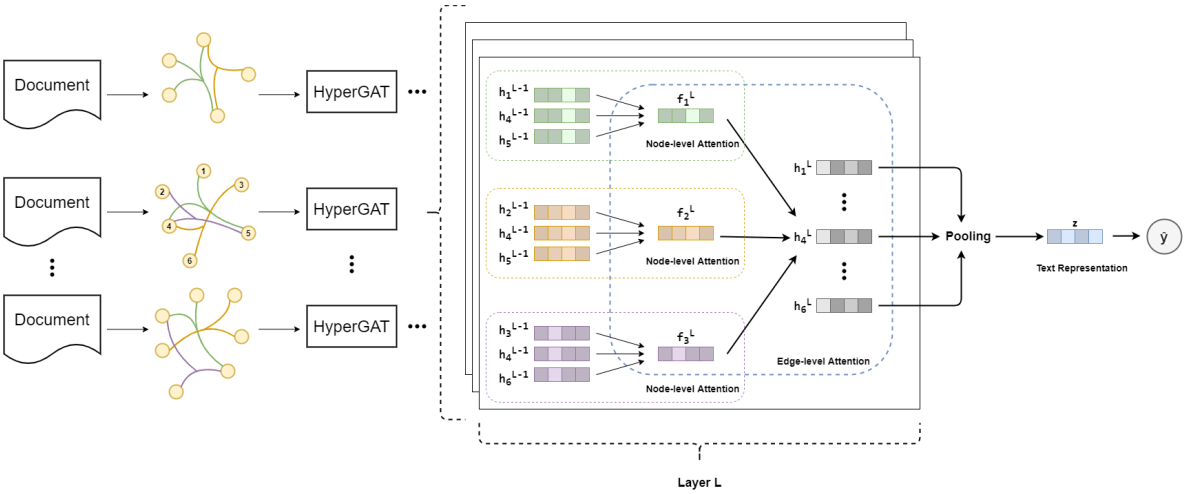


Figure 5.2: High-level view of HyperGAT. Best viewed in color.

### 5.3.2 HyperGAT

In this subsection, we present the HyperGAT network [6] for text classification. Figure 5.2 illustrates a high-level view of the network. The details are described as follows.

**Hypergraph construction.** Hypergraph is a special type of graph, in which its hyperedges can connect more than 2 nodes. For each document, a hypergraph is constructed where the nodes are the words from the document and the hyperedges consists of sequential hyperedges and semantic hyperedges. In [6], 2 ways of constructing sequential hyperedges are mentioned. The first way is to slide a fixed-size sliding window over a document, and connect words inside the window by a hyperedge each time. The second way leverages the document structural information by using a hyperedge to connect all words in each sentence in the document. Semantic hyperedges are used to capture the semantic context in a document. Specifically, HyperGAT uses LDA [1] to mine latent topics in a document, then takes the top N words of each latent topic to represent that latent topic. Next, words in the document that belong to any latent topic (i.e., in the list of words that represents a latent topic) are connected by a hyperedge.

**Dual attention mechanism.** HyperGAT uses dual attention mechanism which consists of node-level attention and edge-level attention. Specifically, hyperedge representations are learned by node-level attention (i.e., for each hyperedge, its representation is learned based on attention mechanism on the nodes connected to it), and node representations are learned by edge-level attention (i.e., for each node, its representation is learned based on attention mechanism on the hyperedges connected to it). The learned node representations of the previous layer will be fed to the next layer as inputs. The document representation is obtained via pooling the node representations in the last layer.

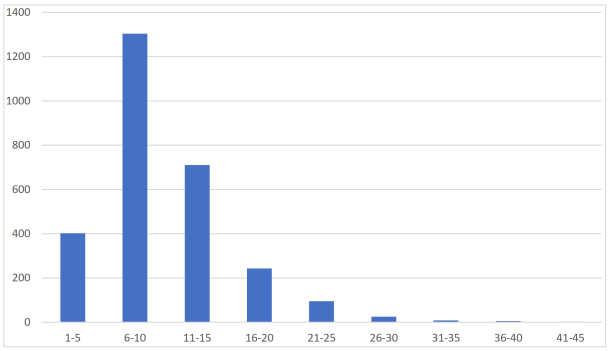


Figure 5.3: Sentence length frequency distribution.

### 5.3.3 Investigation of the contribution of malware-attribute hyperedges

As mentioned above, HyperGAT employs sequential hyperedges and semantic hyperedges. Regarding sequential hyperedge construction, for each document, all words in each sentence are connected by a hyperedge. Semantic hyperedges are supposed to capture topic-related high-order relations between words in a document. To this end, top K words with the highest probabilities of each latent topic mined from the document by LDA [1] are connected by a hyperedge. For the task of malware action classification, we keep the semantic hyperedges and modify the method to construct sequential hyperedges. Specifically, in this task, we consider sentences instead of documents (see Figure 5.3 for sentence length frequency distribution). Therefore, as a way to leverage sequential context, we adopt a fixed-size sliding window to acquire global word co-occurrence. In addition to the above-mentioned hyperedges, we also construct malware-attribute hyperedges for each document hypergraph following [35]. Specifically, because the relevant words carry important information to identifying attribute label, those relevant words – called MAEC keywords – are collected for each attribute label. Besides, sentences of the same attribute label are formed as a document and top N most informative words (ranked by tf-idf scores) of this document are considered the document keywords corresponding to the attribute label. MAEC keywords and document keywords of each attribute label are then combined to get the attribute keywords of this attribute label. In the hypergraph construction phase, for each malware-related sentence, in addition to sequential and semantic hyperedges, the words which appear in any attribute keyword set are connected by a hyperedge, which we call malware-attribute hyperedge.

### 5.3.4 Validation of the superiority of sequential hyperedge to sequential normal-edge

As aforementioned, the sequential hyperedges are constructed by adopting a fixed-size sliding window. To validate the superiority of sequential hyperedge to sequential normal-edge, we set the window size equals 2 for sequential normal-edges and window size larger

than 2 for sequential hyperedges.

## 5.4 Experimental Results

**Data split and allocation.** In 39 annotated documents of *MalwareTextDB* dataset, there are 2,975 malware-related sentences, each of them contains one malware action. As proposed by the dataset authors [21], we split the dataset into 3 parts for training, validation and testing with the ratio 60% : 20% : 20%, respectively. It results in 23 documents for training, 8 documents for validation and 8 documents for testing. We also conduct each experiment 5 times with random data split and allocation. After that, the average F-scores are reported.

**Classifiers.** As aforementioned, a malware action (which corresponds to a malware-related sentence) can be classified into more than one category, but for each of those categories, there is at most one attribute label. Therefore, for each category (ActionName, Capability, StrategicObjectives or TacticalObjectives), we build an *n+1*-way classifier, where n is the number of attribute labels in the category and 1 is the "other" label, meaning the malware action is not classified into any attribute label of the category.

**Experimental settings.** Because the sentence lengths mainly distributed in the range from 1 to 15 words and many of them are from 1 to 5 (see Figure 5.3), we set the window size equals 3 for the sequential hyperedges. All the experiments are run with 10 epochs with early stopping. We keep the parameters in HyperGAT [6]: L2 regularization = $10^{-6}$, dropout rate = 0.3, top N keywords from each latent topic with N = 10, and the number of latent topic equals the number of attribute labels.

### 5.4.1 Investigation of the contribution of malware-attribute hyperedges

Table 5.1 shows the experimental results of the classifiers with and without malware-attribute hyperedges. The average F-scores of classifiers with malware-attribute hyperedges consistently surpass the average F-scores of classifiers without malware-attribute hyperedges, suggesting that malware-attribute hyperedges contribute meaningful features that HyperGAT can exploit. For example, the TacticalObjectives classifier with malware-attribute hyperedges classifies the sentence "Malicious apps steal the passwords if they can read the contents on clipboard" into the attribute label "DataTheft-steal_web/network_credential" while the TacticalObjectives classifier without malware-attribute hyperedges classifies it into "other". The different if that there is an malware-attribute hyperedge connects two non-consecutive words "steal" and "passwords", which seems to be a high-value feature for classification (although there is a sequential hyperedge which connects 3 consecutive words "steal", "the" and "passwords", this hyperedge seems to be noisier than the hyperedge that connects 2 non-consecutive words "steal" and "passwords"). It indicates that the ability of connecting non-consecutive attribute keywords of malware-attribute hyperedges may strengthen the classifiers.

Table 5.1: Experimental results of the classifiers with and without malware-attribute (MA) hyperedges.

| Average F-scores | ActionName | Capability | StrategicObj | TacticalObj |
|---|---|---|---|---|
| w/o MA hyperedges | 70.04 | 51.56 | 39.49 | 39.53 |
| with MA hyperedges | **71.42** | **54.34** | **41.51** | **41.27** |

Table 5.2: Experimental results of the classifiers with sequential hyperedges and sequential normal-edges

| Average F-scores | ActionName | Capability | StrategicObj | TacticalObj |
|---|---|---|---|---|
| with normal-edges | 70.18 | 52.70 | 39.83 | 39.95 |
| with hyperedges | **71.42** | **54.34** | **41.51** | **41.27** |

### 5.4.2 Validation of the superiority of sequential hyperedge to sequential normal-edge

Table 5.2 shows the experimental results of the classifiers with sequential hyperedges or sequential normal-edges. The average F-scores of classifiers with sequential hyperedges consistently surpass the average F-scores of classifiers with sequential normal-edges, suggesting that sequential hyperedges contribute more meaningful features that HyperGAT can exploit than sequential normal-edges. For example, the TacticalObjectives classifier with sequential hyperedges classifies the sentence "the malware logging their keystrokes" into the attribute label "Spying-capture_keyboard_input" while the TacticalObjectives classifier sequential normal-edges classifies it into "other". The different if that the earlier classifier can connect 3 words "logging", "their", "keystrokes" by a hyperedge to make a more meaningful connections than the latter classifier did: connect "logging" with "their", and connect "their" with "keystrokes". It indicates that in cases where a useful feature is formed by the connections of many words, hyperedges is obviously surpass normal-edges.

## 5.5 Conclusion

In conclusion, in this chapter, we demonstrate how the malware-attribute hyperedges can contribute to HyperGAT on the task of malware action classification, and validate the ability of hyperedges to forming useful features.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

In this research, we address the four subtasks of analyzing cybersecurity texts:

- **Subtask 1**: Identify malware-related sentences (i.e., sentences which describe malware actions) from APT reports
- **Subtask 2**: Identify token labels (i.e., Malware Action, Subject of Action, Object of Action, and Modifier of Action) in malware-related sentences
- **Subtask 3**: Identify relation labels (i.e., Subject-Action, Action-Object, Action-Modifier, and Modifier-Object) between tokens
- **Subtask 4**: Classify malware actions into attribute labels.

For subtask 1, two methods are presented for the purpose of exploiting knowledge in an external document (the Attribute Reference Guide) to produce enriched features for sentences. One method employs GATs to obtain a *weak label* for a sentence, where the other produce 444 weights corresponding to 444 attribute labels, to further help representing the sentences. With those features, we gain an improvement of about 9% comparing to the performance of our SVM model without those features, and we achieve the SoTA performance on this task, on the *MalwareTextDBv2.0* dataset.

For subtask 2, we propose an BERT-CRF model for the task of token labelling. With the post-processing phase, which utilizes the our predictions from subtask 1, we achieve the state-of-the-art performance for subtask 2 on the *MalwareTextDBv2.0* dataset.

In subtask 3, we follow the rules designed in [21] to generate relation labels. After that, we address the coreference issue to construct a graph and visualize it.

For subtask 4, we propose to employ a GNN model for the task of malware action classification.

The experiments demonstrate the promising results of neural networks (in general) and graph neural networks (in particular) for the task of analyzing cybersecurity texts.

## 6.2 Future Work

Based on the promising results of this thesis, we would like to further investigate:

- **Subtask 3: Identify relation labels**: The current approach utilizes a rule-based method. This approach has some drawbacks: it requires hand-crafted rules, and is inflexible (as the set of rules is fixed). An neural network approach can be employed for this subtask. One possible direction is to employ supervised learning, where the model has the ability to learn latent features from the annotated data - which may result in a more flexible relation extraction model. In case the annotated data is low-resource, self-labeled techniques [38] should be helpful as they allow a not-fully-converge model (which is being trained on gold data) to make predictions for the purpose of generating *silver* data.

- **Subtask 4: Classify malware actions into attribute labels**: As the annotated data for this subtask is not plentiful, the utilization of the unannotated data (which can be easily crawled from the Internet) should benefit the classification performance. A BERT-like paradigm [14] is proposed which allows pretraining and finetuning for graph-structured data, however, the knowledge transfer phase of this paradigm is not always positive: it may result in negative transfer in cases where the pretraining tasks and the finetuning task are not closely related. We would like to investigate how the graphs can be constructed from cybersecurity (unannotated and annotated) texts that ensures positive transfer.

# Bibliography


[1] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *the Journal of machine Learning research* 3 (2003), pp. 993–1022.

[2] Chris Brew. "Digital Operatives at SemEval-2018 Task 8: Using dependency features for malware NLP". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. 2018, pp. 894–897.

[3] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. "Online Passive-Aggressive Algorithms". In: *J. Mach. Learn. Res.* 7 (Dec. 2006), pp. 551–585. ISSN: 1532-4435.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems*. 2016. URL: https://arxiv.org/abs/1606.09375.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[6] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. "Be More with Less: Hypergraph Attention Networks for Inductive Text Classification". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 4927–4936. DOI: 10.18653/v1/2020.emnlp-main.399. URL: https://www.aclweb.org/anthology/2020.emnlp-main.399.

[7] Ming Ding, Jie Tang, and Jie Zhang. "Semi-supervised learning on graphs with generative adversarial nets". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 913–922.

[8] Kien Do, Truyen Tran, and Svetha Venkatesh. "Graph transformation policy network for chemical reaction prediction". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 750–760.

[9] Mingming Fu, Xuemin Zhao, and Yonghong Yan. "HCCL at SemEval-2018 Task 8: An End-to-End System for Sequence Labeling from Cybersecurity Reports". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. 2018, pp. 874–877.

[10] Aleksa Gordić. *pytorch-GAT*. `https://github.com/gordicaleksa/pytorch-GAT`. 2020.

[11] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee. 2013, pp. 6645–6649.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 1025–1035.

[13] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. "Support vector machines". In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. "Strategies for Pre-training Graph Neural Networks". In: *International Conference on Learning Representations*. 2020. URL: `https://openreview.net/forum?id=HJlWWJSFDH`.

[15] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *CoRR* abs/1408.5882 (2014). arXiv: `1408.5882`. URL: `http://arxiv.org/abs/1408.5882`.

[16] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[17] Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. "Malware attribute enumeration and characterization". In: *The MITRE Corporation [online, accessed May. 20, 2021]* (2011).

[18] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 1-55860-778-1. URL: `http://dl.acm.org/citation.cfm?id=645530.655813`.

[19] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *International Conference on Learning Representations*. 2020. URL: `https://openreview.net/forum?id=H1eA7AEtvS`.

[20] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. "Graph classification using structural attention". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 1666–1674.

[21] Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. "MalwareTextDB: A Database for Annotated Malware Articles". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1557–1567. DOI: 10.18653/v1/P17-1143. URL: https://www.aclweb.org/anthology/P17-1143.

[22] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. "Tensor Graph Convolutional Networks for Text Classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 8409–8416. DOI: 10.1609/aaai.v34i05.6359. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6359.

[23] Yijin Lium, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. "GCDT: A Global Context Enhanced Deep Transition Architecture for Sequence Labeling". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[24] Pablo Loyola, Kugamoorthy Gajananan, Yuji Watanabe, and Fumiko Satoh. "Villani at SemEval-2018 Task 8: Semantic Extraction from Cybersecurity Reports using Representation Learning". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. 2018, pp. 885–889.

[25] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166.

[26] Chunping Ma, Huafei Zheng, Pengjun Xie, Chen Li, Linlin Li, and Luo Si. "DM_NLP at SemEval-2018 Task 8: neural sequence labeling with linguistic features". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. 2018, pp. 707–711.

[27] Xuezhe Ma and Eduard Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1064–1074. DOI: 10.18653/v1/P16-1101. URL: https://aclanthology.org/P16-1101.

[28] R Manikandan, Krishna Madgula, and Snehanshu Saha. "TeamDL at SemEval-2018 task 8: Cybersecurity text analysis using convolutional neural network and conditional random fields". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. 2018, pp. 868–873.

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119.

[30] Federico Monti, Michael Bronstein, and Xavier Bresson. "Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[31] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proceedings of NAACL-HLT 2019: Demonstrations*. 2019.

[32] Ankur Padia, Arpita Roy, Taneeya W Satyapanich, Francis Ferraro, Shimei Pan, Youngja Park, Anupam Joshi, and Tim Finin. "UMBC at SemEval-2018 Task 8: Understanding text about malware". In: *UMBC Computer Science and Electrical Engineering Department* (2018).

[33] 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)*. 2014, pp. 1532–1543.

[34] Peter Phandi, Amila Silva, and Wei Lu. "SemEval-2018 task 8: Semantic extraction from CybersecUrity REports using natural language processing (SecureNLP)". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. 2018, pp. 697–706.

[35] Arpita Roy, Youngja Park, and Shimei Pan. "Predicting Malware Attributes from Cybersecurity Texts". 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2857–2861. DOI: 10.18653/v1/N19-1293. URL: https://www.aclweb.org/anthology/N19-1293.

[36] Utpal Kumar Sikdar, Biswanath Barik, and Björn Gambäck. "Flytxt_NTNU at SemEval-2018 Task 8: Identifying and Classifying Malware Text Using Conditional Random Fields and Naive Bayes Classifiers". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. 2018, pp. 890–893.

[37] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. "Learning deep representations for graph clustering". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

[38] Isaac Triguero, Salvador Garcıa, and Francisco Herrera. "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study". In: *Knowledge and Information systems* 42.2 (2015), pp. 245–284.

[39] Ke Tu, Peng Cui, Xiao Wang, Philip S Yu, and Wenwu Zhu. "Deep recursive network embedding with regular equivalence". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 2357–2366.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks". In: *International Conference on Learning Representations* (2018). accepted as poster. URL: https://openreview.net/forum?id=rJXMpikCZ.

[41] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. "Deep Graph Infomax". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rklz9iAcKQ.

[42] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. "Graphgan: Graph representation learning with generative adversarial nets". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[43] Thomas Wolf et al. *Transformers: State-of-the-art Natural Language Processing*. 2019. arXiv: 1910.03771 [cs.CL].

[44] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. "Simplifying Graph Convolutional Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6861–6871. URL: http://proceedings.mlr.press/v97/wu19e.html.

[45] Liang Yao, Chengsheng Mao, and Yuan Luo. "Graph convolutional networks for text classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 7370–7377.

[46] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. "Graphrnn: Generating realistic graphs with deep auto-regressive models". In: *International conference on machine learning*. PMLR. 2018, pp. 5708–5717.

[47] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. "Learning deep network representations with adversarially regularized autoencoders". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 2663–2671.

# Papers and Awards

## Papers

[1] Chau Nguyen, Vu Tran, and Le Minh Nguyen. "Enrichment of Features for Malware-RelatedSentence Classification using External Knowledge". In: *Proceedings of the 33rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE. 2021.

## Papers which are not in the thesis

[1] Ha-Thanh Nguyen, Phuong Minh Nguyen, Hai-Yen Thi Vuong, Quan Minh Bui, Chau Minh Nguyen, Binh Tran Dang, Vu Tran, Minh Le Nguyen, and Ken Satoh. "JNLP Team: Deep Learning Approaches for Legal ProcessingTasks in COLIEE 2021". In: *Proceedings of the Eighth International Competition on Legal Information Extraction/Entailment (COLIEE 2021)*. 2021.

[2] Ha-Thanh Nguyen, Hai-Yen Thi Vuong, Phuong Minh Nguyen, Binh Tran Dang, Quan Minh Bui, Sinh Trong Vu, Chau Minh Nguyen, Vu Tran, Ken Satoh, and Minh Le Nguyen. "JNLP Team: Deep Learning for Legal Processing in COLIEE 2020". In: *Proceedings of the Seventh International Competition on Legal Information Extraction/Entailment (COLIEE 2020)*. 2020.

[3] Vu Tran, Van-Hien Tran, Phuong Nguyen, Chau Nguyen, Ken Satoh, Yuji Matsumoto, and Minh Nguyen. "CovRelex: A COVID-19 Retrieval System with Relation Extraction". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 2021, pp. 24–31.

## Awards

[1] Autumn 2019 Monbukagakusho Honors Scholarship, October 2019.