symmetry

MDPI

Article

K-CTIAA: Automatic Analysis of Cyber Threat Intelligence Based on a Knowledge Graph

Zong-Xun Li [1,*], Yu-Jun Li [1], Yi-Wei Liu [1], Cheng Liu [1,2] and Nan-Xin Zhou [1]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2] Science and Technology on Communication Security Laboratory, The 30th Research Institute of China Electronics Technology Group Corporation, Chengdu 610041, China
* Correspondence: 202021080328@std.uestc.edu.cn

Abstract: Cyber threat intelligence (CTI) sharing has gradually become an important means of dealing with security threats. Considering the growth of cyber threat intelligence, the quick analysis of threats has become a hot topic at present. Researchers have proposed some machine learning and deep learning models to automatically analyze these immense amounts of cyber threat intelligence. However, due to a large amount of network security terminology in CTI, these models based on open-domain corpus perform poorly in the CTI automatic analysis task. To address this problem, we propose an automatic CTI analysis method named K-CTIAA, which can extract threat actions from unstructured CTI by pre-trained models and knowledge graphs. First, the related knowledge in knowledge graphs will be supplemented to the corresponding position in CTI through knowledge query and knowledge insertion, which help the pre-trained model understand the semantics of network security terms and extract threat actions. Second, K-CTIAA reduces the adverse effects of knowledge insertion, usually called the knowledge noise problem, by introducing a visibility matrix and modifying the calculation formula of the self-attention. Third, K-CTIAA maps corresponding countermeasures by using digital artifacts, which can provide some feasible suggestions to prevent attacks. In the test data set, the F1 score of K-CTIAA reaches 0.941. The experimental results show that K-CTIAA can improve the performance of automatic threat intelligence analysis and it has certain significance for dealing with security threats.

Keywords: cyber threat intelligence; pre-trained model; threat action extraction; cyber security knowledge graph

Citation: Li, Z.-X.; Li, Y.-J.; Liu, Y.-W.; Liu, C.; Zhou, N.-X. K-CTIAA: Automatic Analysis of Cyber Threat Intelligence Based on a Knowledge Graph. *Symmetry* **2023**, *15*, 337. https://doi.org/10.3390/sym15020337

Academic Editor: Christos Volos

Received: 29 December 2022
Revised: 14 January 2023
Accepted: 17 January 2023
Published: 25 January 2023

1. Introduction

As communication technology improves rapidly, a growing number of critical infrastructures in our daily life rely on Internet access [1]. However, people are relatively backward in preventing cyber threat actions, and these cyber threat actions against infrastructure will seriously affect people's daily life [2]. How to prevent cyber threats has become a hot topic at present. Recently, cyber threat intelligence sharing has gradually become an important means of dealing with APT attacks. With cyber threat intelligence sharing, security experts can quickly capture the steps of attack and find the most effective preventive solution [3]. However, the infinite growth of cyber threat reports consumes the limited energy of security experts, which makes it difficult to conduct a timely analysis of each report. Therefore, this paper proposes K-CTIAA, a cyber threat intelligence analysis method based on a pre-trained model and knowledge graphs, which implements the automatic extraction of key information in cyber threat intelligence and helps security experts to be liberated from dozens of pages of cyber threat intelligence. The pre-trained model learns general language knowledge through self-supervision in a large-scale corpus, and this general knowledge can be applied to a wide variety of downstream tasks. In

this way, instead of training a model from scratch with a large amount of data, it can solve new problems with very few samples [4]. The pre-trained language representation model has achieved excellent results in many tasks in natural language processing, and the K-CTIAA experimental results show that it also performs well in the field of network threat intelligence analysis.

Due to the abundance of cyber security terminology in cyber threat intelligence, pre-trained models based on open domain corpus cannot accurately parse the semantics of these specialized terms, which may affect the performance of the cyber threat intelligence analysis task [5]. To address this problem, K-CTIAA looks up the related knowledge of professional terms in the knowledge graph through knowledge query and inputs this knowledge into the pre-trained model. The additional input knowledge can help the pre-trained model understand the semantics of network security terms. Additionally, to allow security experts to quickly respond to incoming cyber threat actions, K-CTIAA provides security experts with countermeasures based on relevant knowledge graphs [6].

Researchers have achieved some results for the automatic analysis of cyber threat intelligence, but these studies rarely consider the professional terms that appear in cyber threat intelligence [7]. Some methods based on machine learning and rules require the participation of experts, which can greatly reduce the efficiency of automated threat intelligence analysis. In addition, some methods based on deep learning cannot understand proper nouns in threat intelligence due to their imperfect semantic parsing, and the accuracy of extraction is low. The main contributions of this paper are described as follows:

- This paper proposes a cyber threat intelligence analysis method by inserting knowledge into the pre-trained model, which can parse the semantic information of network security terms in sentences and extract threat actions.
- This paper introduces a visibility matrix and modifies the calculation formula of the self-attention to reduce the negative impact of knowledge insertion, which is usually called the knowledge noise problem. The performance of the proposed measures is verified in experiments.
- This paper uses the mapping relationship between ATT&CK [8] and D3fend [6] (network security knowledge graphs) to provide countermeasures for the extracted threat actions, which can help security experts quickly respond to upcoming threats.

The rest of this paper is organized as follows: Section 2 discusses the details of related works and background. Section 3 introduces the overall architecture of the CTI analysis system and discusses the implementation details of each part in detail. Section 4 displays and analyzes the experimental results, and verifies the effectiveness of our method by comparing the experimental results. Section 5 discusses several key issues about K-CTIAA, while Section 6 contains future work and conclusions.

2. Related Works

2.1. Cybersecurity Threat Intelligence

Cybersecurity threat intelligence analysis is a topic of ongoing interest, and some research has been done in this area. Niakanlahiji et al. [9] and Husari et al. [10] applied syntactic parsing to extract subject–verb–object (SVO) structure from each sentence in CTI, and extracted threat actions based on the SVO structures. Fujii et al. [11] extracted key information from cyber threat intelligence by using named entities and combining them. However, these methods perform poorly in complicated sentences since they ignore the semantic information in the sentences. Noor et al. [12] presented a method based on semantic similarity to extract actions. However, due to the complexity of the calculation of the latent semantic indexing method, when this method is applied to a data set with a large amount of data, it takes a long time to calculate. Husari et al. [13] used entropy and mutual information to extract threat actions. Ramnani et al. [14] proposed a semi-automated method to extract critical information from security documents using pattern matching. However, this method requires manual participation and is inefficient. Satvat et al. [5] used the pre-trained model BERT for threat action extraction and achieved good results,

but the method ignores the terminology in CTI. Sun et al. [15] proposed an automatic approach to generate the CTI records based on multi-type open source threat intelligence management platforms (OSTIPs), combing the NLP method, machine learning method, and cybersecurity threat intelligence knowledge. Preuveneers et al. [16] designed and evaluated a CTI solution that complements the attribute and tagging-based sharing of indicators of compromise with machine learning models for collaborative threat detection.

2.2. Cyber Security Knowledge Graph

A knowledge graph is essentially a semantic network that reveals the relationships between entities [17]. Computers have always faced the dilemma of not obtaining the semantic information of web texts. Although artificial intelligence has made great strides in recent years, even surpassing human performance in certain tasks, it remains satisfactory in some professional fields. The inability of machines to attain the meaning behind professional terms is one of the important reasons for this problem, we need to model the descriptive thing, fill its properties, and expand its connection to other things. In cyber threat intelligence analysis tasks, we also face this dilemma. Therefore, we introduce network security knowledge graphs to solve this problem.

Recently, researchers have conducted extensive research on cyber security knowledge graphs. Herzog et al. [18] created a detailed model of information security, including threat, asset, countermeasure classes and relationships among them. Jia et al. [19] used machine learning to take entities and build an ontology to obtain a cyber security knowledge graph. Syed et al. [20] built a cybersecurity ontology that integrates heterogeneous data and knowledge schemas from different cyber security systems and the most commonly used cybersecurity standards for information sharing and exchange. Noel et al. [21] built CyGraph, a cybersecurity knowledge graph that integrates isolated data and events into a holistic picture for decision support and situational awareness. Ren et al. [22] designed an Advanced Persistent Threat (APT) knowledge extraction algorithm for completing and updating the knowledge graph using deep learning and expert knowledge. Li et al. [23] constructed a standardized network security ontology based on knowledge graphs. Hooi et al. [24] proposed a framework to create a knowledge graph of threat actors by building an ontology of threat actors and a named entity recognition system to extract cybersecurity-related entities. Many researchers have open-sourced their knowledge graphs on GitHub [25,26]. However, these knowledge graphs have problems with incomplete information and inconsistent formats, and they need to be preprocessed before they are used.

2.3. Pre-Trained Model Combined with a Knowledge Graph

Pre-trained language representation models such as BERT [27] can obtain common language representations from large corpora. The BERT model has achieved good performance in various tasks of natural language processing. At the same time, some researchers are applying BERT to professional fields, such as military and agriculture [28,29].

However, due to the lack of specialized knowledge in the general corpus, the model has a poor understanding of the semantic information of proper nouns that rarely appear in the general corpus. To address this issue, some researchers combine knowledge graphs with pre-trained models to annotate these difficult terms. As a result, the pre-trained model can obtain the semantics of these terms more accurately. K-BERT [30] proposes a knowledge-supported language representation model, which is compatible with BERT and can combine related knowledge to solve the situation where the pre-trained model does not perform well in a specific domain. K-BERT inserts knowledge into the input in the form of branches and changes the model structure to solve the knowledge noise problem. K-BERT significantly outperforms BERT not only on domain-specific tasks, but also on open-domain tasks.

Some scholars also use other methods to integrate knowledge maps into pre-trained models. These methods can be divided into three types: implicit fusion, explicit fusion, and joint learning of a knowledge graph and a pre-trained model [31].

- Implicit fusion—use embedding to fuse inside the model

Implicit fusion is a relatively direct embedding fusion method. This type of method is based on some KGE (Knowledge Graph Embedding, the most used is Translating Embedding) algorithms to obtain the embeddings of entities and relationships in the knowledge graph, and modify the pre-trained model structure for these embeddings. This type of method constructs the semantic space of the pre-trained model according to the relationship between entities in the knowledge. Representative papers on such methods include ERNIE [32].

- Explicit fusion—fusion without changing the model structure

Compared with the embedding-based combination described above, this idea is more direct: the entities and relationships are directly input into the pre-trained model in the form of tokens after some transformations. The difference from the above methods is that the knowledge is directly fed into the model. Representative papers on such methods include K-BERT [30].

- Joint learning of a knowledge graph and a pre-trained model

In addition to these two methods, some researchers have tried a new approach. This kind of method is not satisfied by simply enhancing the knowledge map to the pre-trained model, but by processing the knowledge map representation learning and knowledge-oriented natural language understanding together. Representative papers on such methods include KEPLER [33].

3. Methodology

3.1. K-CTIAA Architecture

The structure of the cyber threat intelligence automatic analysis model KCTIAA based on the pre-trained model and knowledge graph proposed in this paper is shown in Figure 1. K-CTIAA can be divided into three modules, named the preprocessing module, the cyber threat intelligence analysis module, and the countermeasures module, respectively.

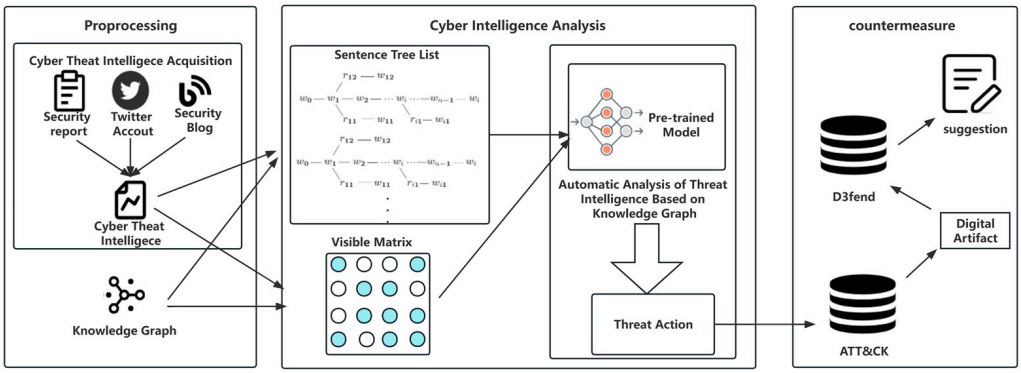

Figure 1. Framework of K-CTIAA.

In the preprocessing module, as shown in Figure 1, cyber threat intelligence from security companies and related Twitter blogs is automatically crawled by the web crawler tool and saved locally in PDF format. These local PDF files are then converted into model-readable text format by tools such as PDFMiner [34]. In addition, converting some specialized terms in CTI to a more general representation is another work in the preprocessing stage. The purpose of this step is to unify the form of some special nouns. For example, nouns about IP addresses often appear in CTI. They will exist in various forms, such as 192.0.0.112, 207.46.13.174, etc. Despite their various formats, when inserting knowledge, they should all match the term IP address in the knowledge graph. Thus, before the knowledge query step, we need to replace these synonyms with predefined nouns in the knowledge graph according to specific rules. In this way, knowledge can be

queried more conveniently. In this module, the construction of knowledge graphs is also one of the tasks to be completed. In K-CTIAA, we use the knowledge graphs proposed by some previous researchers and convert them into triples.

The network threat intelligence analysis module first searches knowledge related to entities through knowledge query, which is usually stored in the form of triples. The knowledge obtained from the query is then inserted into the original input statement in the form of tree structure branches through knowledge insertion. Since the model can only accept input with linear structure, as shown in Figure 1, the network threat intelligence analysis module converts the tree input into linear input that can be read by the model according to certain rules. At the same time, in order to reduce the knowledge noise problem caused by inserting knowledge, the cyber threat intelligence analysis module constructs a visible matrix according to the structure of the sentence tree and adds the visible matrix to the calculation formula of self-attention. The construction of the visible matrix is described in Section 3.3. Finally, the threat actions are obtained by the BERT model with modified calculation rules. The detailed process of this module will be demonstrated in Sections 3.2 and 3.3.

The main purpose of the countermeasure module is to find countermeasures that match the threat actions. This function of k-CTIAA relies on some efforts of the knowledge graph D3fend. D3fend provides a series of countermeasures, which correspond to threat actions in ATT&CK. As shown in Figure 1, K-CTIAA only needs to map the threatening behavior to ATT&CK and then analyze the digital artifacts between ATT&CK and d3defend to obtain the corresponding countermeasures. The detailed process of this module will be introduced in Section 3.5.

3.2. Knowledge Query and Knowledge Insertion

Before knowledge query, we need to build a network security knowledge graph. In Section 2.2, we discuss in detail the related research in the field of network security knowledge graphs. Many researchers have proposed the construction technology of network security knowledge graphs. The knowledge graph used in K-CTIAA is based on previous research [6,8] and some open-sourced knowledge graphs [25,26]. However, the knowledge maps constructed by different researchers are stored in different forms. We wrote a program that reads these knowledge maps and converts them into a unified format. In K-CTIAA, knowledge graphs are organized and stored in the form of entity–relation–entity triples.

How knowledge graphs are applied to pre-trained models has always been a hot topic of research; Section 2.3 introduces the related research work in detail. Considering the replaceability and extensibility of the model, K-CTIAA chooses the method that does not need to change the model structure. Please refer to Section 2.2 for the detailed introduction of this method. First, K-CTIAA inserts knowledge into cyber threat intelligence to construct a sentence tree with knowledge branches, and then transforms the sentence tree with cyber threat intelligence and knowledge into the BERT model. Compared with other methods, the way of inserting knowledge on the input structure makes the BERT model require little modification, and we can easily replace the BERT model with other pre-trained models. In addition, K-CTIAA can also be quickly applied to other fields after changing the knowledge graph.

Specifically, the process of knowledge fusion is divided into two steps: knowledge query and knowledge insertion. Knowledge query refers to identifying knowledge-related entities in a sentence and finding all knowledge associated with these entities. The method used in this step is keyword matching. First, we need to complete a named entity recognition task to mark all the entities in the sentence. For each entity in the sentence, we need to find matching triples in the knowledge graph. If the entity can match the corresponding triple in the knowledge graph, then the triple is called a piece of knowledge of the entity. We need to note that an entity may correspond to multiple knowledge.

Knowledge insertion refers to inserting relevant knowledge into a sentence and affecting the semantics of the original sentence as little as possible. In fact, knowledge insertion

is a complicated matter. We need to complete the embedding of knowledge semantics without affecting the sentence structure. In K-CTIAA, we use the method of constructing a sentence tree to insert knowledge into the sentence, and then convert the input of the tree structure into the input of the linear structure through specific rules, and modify the calculation method of the model to limit the interference of too much knowledge. All triples matched in the knowledge query step will be inserted into the sentence as branches. As shown in Figure 2, "OceanLotus" matches the relevant entities in the knowledge graph, and the corresponding triple "OceanLotus is group" is inserted into the original sentence as a branch. "Phishing" will be matched to multiple triples which will all be inserted into the sentence as branches.

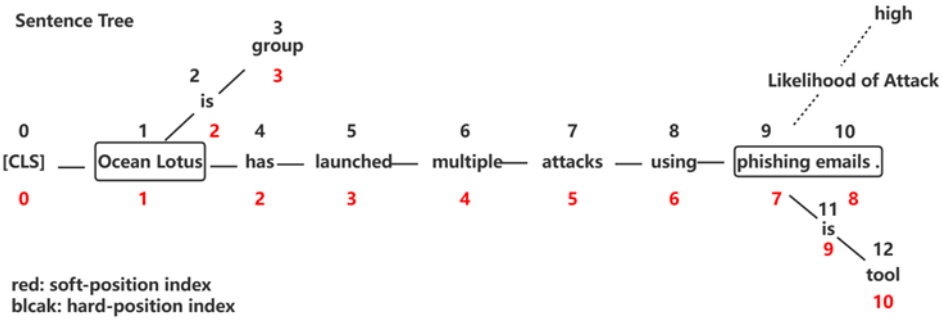

Figure 2. An example of knowledge query and knowledge insertion.

3.3. Cyber Threat Intelligence Analysis with BERT

In Section 3.2, we obtained a knowledge-integrated sentence tree through knowledge query and knowledge insertion, but the input of the pre-trained model is linear, and the tree-structured sentence tree cannot be directly input into the model. How to convert the input of a tree-shaped structure into a linear structure is the first problem we need to solve.

A concept of hard position coding and soft position coding is introduced to solve this problem. The hard position encoding is the unique encoding of each token according to the depth-first and branch-first rules. According to the number of hard position coding, we can convert the input tree into linear input, but this conversion will cause the loss of the input structure information. Therefore, we put forward a concept of soft position coding. The soft position encoding is the relative encoding according to the different depths of the token in the sentence tree, soft location coding is mainly used to store the input structure information. As shown in Figure 2, the red number under each token represents the soft position of the token, and the black number under each token represents the hard position of the token. The soft position index is fed into the model as location embeddings in the BERT model. In this way, BERT model can obtain the structure information of sentence tree indirectly. The tokens in the sentence are input into the token embedding of the model in the order of hard position encoding. As shown in Figure 3, the input of the model will be obtained by adding three parts, which are token embeddings, segment embeddings, soft position embeddings. The token embedding is the encoding of each token in the input. The segment embedding is the sentence to which the token belongs (this is usually applicable when there are multiple sentences in the input, such as the implication judgment task). The position embedding represents the position of the token in the input. Compared with the BERT model, K-CTIAA replaced the position embedding of the BERT model with soft position embedding, which is the red part in Figure 3.

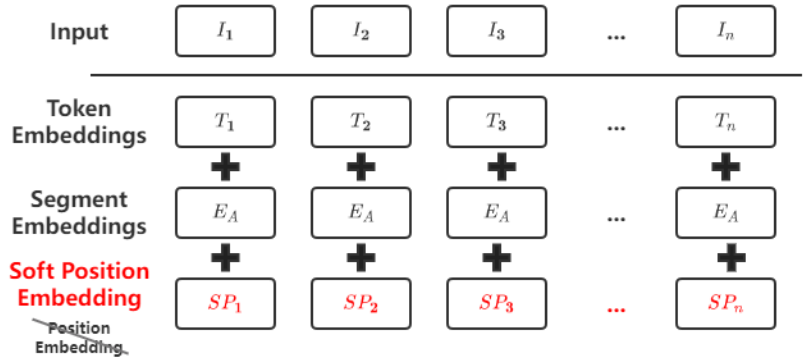

Figure 3. Input structure of the model.

Using only soft positional embedding instead of positional embedding will result in some issues, such as knowledge noise, which refers to the adverse impact of the inserted knowledge on the semantics of the original sentence. First, the structure of the sentence cannot be completely restored from the soft position, the whole sentence will become confusing and unreadable, which will make the sentence lose its original semantics. Moreover, the original sentence and the inserted knowledge become indistinguishable after inserting the knowledge; their role in both input and computation is the same. This is fatal to the model, because the insertion of knowledge is more of a semantic supplement. If the knowledge is indistinguishable from the original sentence, it will seriously affect the results of subsequent task. For example, in Figure 2, the soft positions of tokens "is" and "has" are also 2. If only soft positions are used to embed the input into the model, the model will not be able to distinguish which one is the clause and which one is the main sentence. If we need to extract the subject–verb–object structure of this sentence, the model will not be able to correctly choose the verb of the original sentence. At the same time, the inserted knowledge will also confuse the semantics of the original sentence, making the original sentence unreadable. If we expand the sentence in the order of input, the entire sentence will not conform to the grammatical and semantic knowledge learned by the pre-trained model during the pre-trained step. Therefore, we need to do something to mitigate the negative effects of inserting knowledge.

In order to reduce the impact of inserted knowledge on the original sentence, we construct a visibility matrix based on the positional relationship of the sentence tree. The value of this matrix will affect the calculation of the self-attention mechanism. Our goal is to reduce the direct impact of knowledge on other tokens (tokens unrelated to knowledge) in the sentence. The size of the matrix is N $*$ N (N represents the number of tokens in the entire sentence after inserting knowledge). For the convenience of representation, we give some definitions as follows: w_i represents the token whose hard position is "i"; $w_i \leftrightarrow w_j$ indicates that w_i and w_j are on the same branch; $w_i \Omega\, w_j$ indicates that w_i and w_j are not on the same branch.

In the matrix, the original sentence is considered as one branch, and the inserted knowledge is considered as a separate branch, tokens in the same branch are visible to each other, and tokens that are not in the same branch are invisible to each other. If the two tokens are invisible to each other, it means that they should not affect each other during calculation. Under this rule, the input sentence is invisible to the branch of knowledge, and the branch of knowledge is invisible to the input sentence. Only entities in input sentences can receive semantic information about knowledge. The visible matrix draws on the idea of the self-attention mechanism, and the contribution between invisible tokens should be 0 during calculation. The visible matrix is defined as Equation (1). The visible matrix constructed by the above example sentence according to this rule is shown in Figure 4.

$$M_{ij} = \begin{cases} 0 & w_i \leftrightarrow w_j \\ -\infty & w_i \Omega\, w_j \end{cases} \tag{1}$$

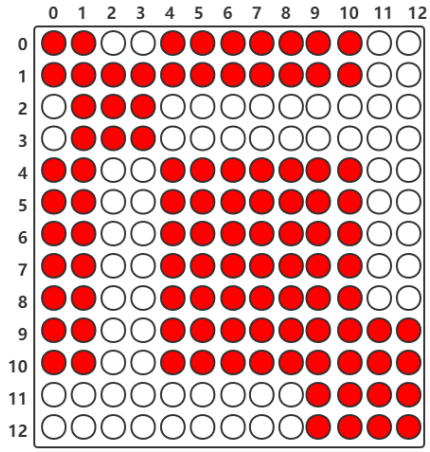

Figure 4. Visible matrix.

Figure 5 shows the application of the visible matrix in self-attention computation. According to the unmodified self-attention calculating rules, all h^{i-1} are visible when calculating a certain h^i. If we add the visible matrix in the calculation rule, the weight of part of the h^{i-1} is set to 0 when calculating the h^i, that is, these h^{i-1} are on invisible state.

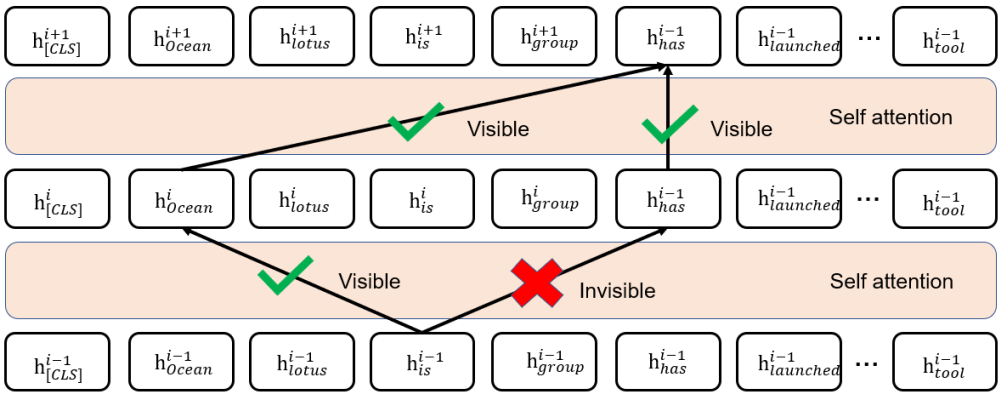

Figure 5. Examples of visible matrix used in self-attention calculations.

According to the visibility matrix, we can easily distinguish the original sentence from the inserted knowledge. For example, the first word of each sentence is the special token [CLS], and this special token is the beginning of the original sentence. Since this token cannot match any triple and cannot be inserted, the set of visible tokens for this special token is all the tokens of the original sentence.

In addition, we can also use the visibility matrix to avoid knowledge noise. By modifying the calculation method of the self-attention mechanism, as shown in Equations (2)–(4), M represents the visible matrix of the sentence. If the two are not on the same branch, the weight after calculating the softmax function is 0, so that the knowledge does not affect the semantic information of the original sentence. If two tokens are on the same branch, the corresponding value in the visible matrix is 0, and the softmax calculation result is exactly the same as the ordinary self-attention mechanism.

$$Q^{i+1}, K^{i+1}, V^{i+1} = h^i W_q, h^i W_k, h^i W_v \tag{2}$$

$$S^{i+1} = \text{softmax}\left(\frac{Q^{i+1}K^{i+1T} + M}{\sqrt{d_k}}\right) \tag{3}$$

$$h^{i+1} = S^{i+1}V^{i+1} \tag{4}$$

3.4. Model Structure Improvements

In Sections 3.1–3.3, we discussed in detail how to integrate the knowledge map into the pre-trained model, but this method still has some problems during the experiment. It can be summarized into the following two points: matching too much knowledge causes the length of the sentence tree to exceed the limits of the model, and knowledge cannot acquire semantics according to context.

Inserting too much knowledge will bring some problems. First, inserting too much knowledge into the original sentence will cause the length of the input to exceed the limits of the model. For example, in the above case, the "pushing" entity will match multiple related knowledge. As shown in Figure 6, many of these knowledge triples are not helpful for the semantic understanding of example sentences. The number of tokens for this knowledge is huge compared to the number of tokens in the original sentence, and some knowledge has no beneficial effect on the semantic understanding of the sentence. If all this knowledge is inserted into the sentence, the length of the sentence will exceed the input limit of the model (the maximum length of the BERT input is limited to 512, which also needs to include [CLS] and [SEP]; the actual available length is only 510). In addition, directly inserting all knowledge may also lead to too messy knowledge.

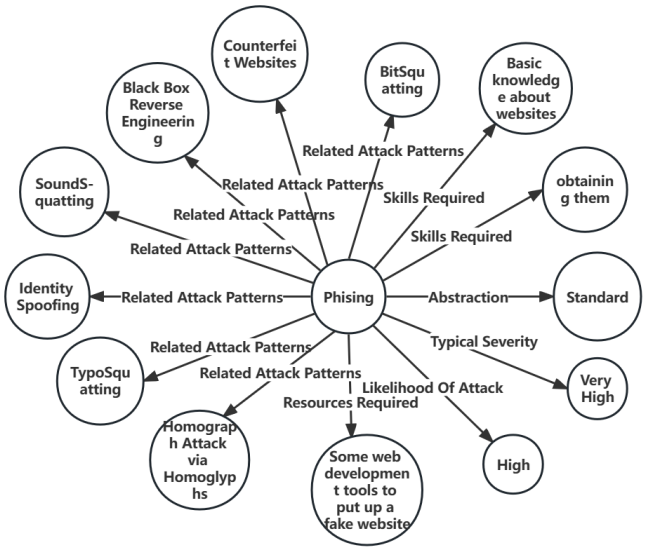

Figure 6. Example of knowledge query result.

Second, K-CTIAA uses triples to organize knowledge graphs, but the knowledge of triples cannot fully reflect the semantic information of knowledge. Simply put, there is a big difference between the structural form of knowledge and the structural form of sentences, which will lead to a poor semantic understanding of knowledge for models trained on the daily corpus. In addition, the semantics of knowledge will also be reflected differently in different contexts, and knowledge needs to be understood in the context.

In response to these two problems, we improved the structure of the model from two aspects. First of all, to solve the problem of inserting too much knowledge, the method we adopt is to sort the relevance of knowledge by Term Frequency-Inverse Document Frequency (TF-IDF), filter the knowledge according to the threshold of relevance score, and only retain the knowledge with semantic relevance higher than the threshold 0.08. After screening, the inserted knowledge of sentences will be greatly reduced. In Section 4, we

will discuss in detail the influence of different threshold values on the number of sentences. The elimination of non-essential knowledge in the sentence tree is represented by the elimination of branches with low semantic relevance. For example, the dashed line in Figure 2 represents the branch that was pruned. We discuss the method for determining the pruning threshold for sentence trees in Section 5.

TF-IDF is a statistical method for evaluating the importance of a word to a document set or a document in a corpus. The basic idea is that the importance of a word increases proportionally to the number of times it appears in the document, but decreases inversely proportional to the frequency it appears in the corpus.

TF (Term Frequency) indicates the frequency of terms appearing in the text. This number is usually normalized (usually the term frequency divided by the total number of words in the article) to prevent it from being biased toward long files. TF is expressed by the formula as follows:

$$TF_{i,j} = \frac{n_{i.j}}{\sum_k n_{k,j}} \tag{5}$$

where $n_{i,j}$ represents the number of occurrences of term t_i in document d_j, and $TF_{i,j}$ represents the frequency of occurrence of term t_i in document d_j.

However, it should be noted that some common words do not have much effect on the theme, but some words that appear less frequently can express the theme of the article, so it is not appropriate to use TF simply. The design of the weight must meet the following: the stronger the ability of a word to predict the topic, the greater the weight, and vice versa. In all statistical articles, some words only appear in a few of them, so such words greatly affect the theme of the article, and the weight of these words should be designed to be larger. IDF does exactly that.

IDF (Inverse Document Frequency) indicates the popularity of keywords. If there are fewer documents containing an entry, a larger IDF means that the entry can distinguish categories. The IDF of a specific term can be obtained by dividing the total number of documents by the number of documents containing the term and then taking the logarithm of the obtained quotient.

$$IDF_i = \log \frac{|D|}{1 + \left|j \ : t_j \ \in \ d_j\right|} \tag{6}$$

where $|D|$ represents the number of all documents, and $\left|j \ : t_j \ \in \ d_j\right|$ represents the number of documents containing t_j.

A high word frequency within a particular document, and a low document frequency of that word in the entire collection of documents, can produce a highly weighted TF-IDF. Therefore, TF-IDF tends to filter out common words and keep important words, expressed as Equation (7).

$$\mathrm{TF} - \mathrm{IDF} \ = \ \mathrm{TF} \times \ \mathrm{IDF} \tag{7}$$

Secondly, to allow the inserted knowledge to also obtain the semantic information of the original sentence, K-CTIAA modifies the construction rules of the visible matrix. Our idea is as follows: we want the inserted knowledge branch to receive semantic information from the main branch, but we want to preserve the restriction of not allowing the main branch to obtain semantics directly from the knowledge branch. This change will transform the visible matrix from a symmetric matrix to an asymmetric matrix, which we believe is valid. We modify the rules of the visible matrix as follows:

- All tokens in the original sentence are visible to the inserted knowledge;
- The inserted knowledge is invisible to the original sentence;
- The inserted knowledge is invisible to each other.

The construction rule of the visible matrix is shown in Equation (8). For the convenience of representation, we give some definitions as follows:

w_i represents the token whose hard position is "i"; $w_i \leftrightarrow w_j$ indicates that w_i and w_j are on the same branch; $w_i \Omega\, w_j$ indicates that w_i and w_j are not on the same branch; W represents the collection of input tokens; $w_i \in W$ indicates that the token character whose hard position is "i" comes from the original input; K represents the collection of inserted knowledge; $w_i \in K$ indicates that the token character whose hard position is "i" comes from the inserted knowledge.

$$M_{ij} = \begin{cases} 0 & w_i \leftrightarrow w_j \\ 0 & w_i \in K \ \ and \ w_j \in W \\ -\infty & others \end{cases} \tag{8}$$

In order to more clearly demonstrate our modification of the visible matrix construction rules, we use the visible matrix in the previous section as an example. In the sentence tree constructed in Section 3.3, under the new visible matrix construction rule, the transformation of the visible matrix is shown in Figure 7.

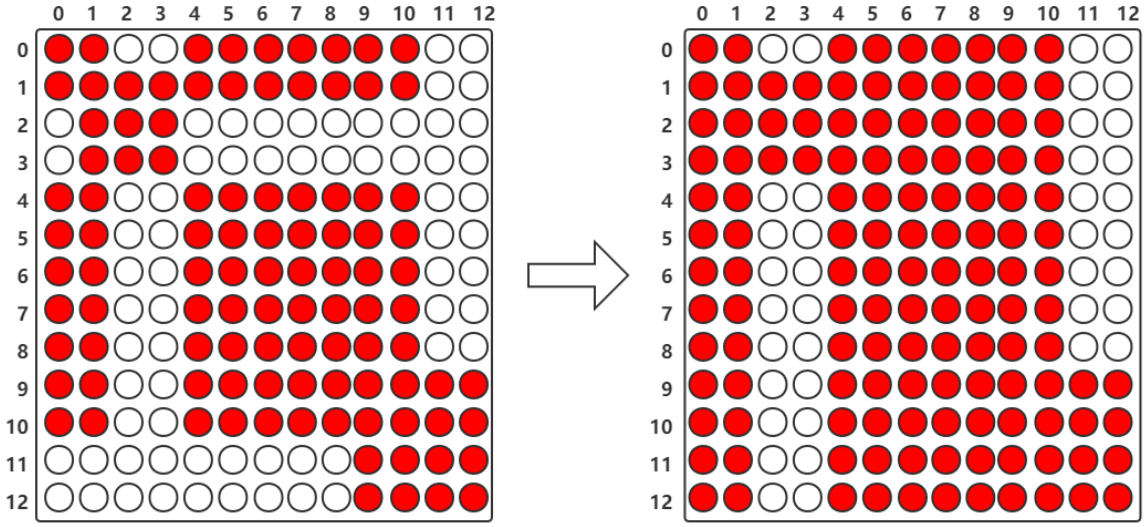

Figure 7. An example of the transformation of the visible matrix definition rules.

3.5. Countermeasure

The countermeasures module is designed to advise security experts on how to respond quickly to upcoming cyber threat actions. However, due to the continuous development of cyber threats, these recommendations cannot completely replace security experts; therefore, the knowledge map of K-CTIAA needs to be updated constantly. The module was implemented by a mapping between D3fend and ATT&CK. D3FEND is a knowledge graph about cybersecurity countermeasure technologies released by MITRE Corporation. This knowledge map collects some strategies to deal with threats and classifies them according to attack techniques. As shown in Figure 8, D3fend uses digital artifacts to map to ATT&CK offensive techniques. This allows us to find the corresponding countermeasure through the mapping relationship after extracting the attack technique. We build a threat action ontology by leveraging MITRE's ATT&CK and CAPEC efforts. Then, K-CTIAA matches the threat information output by the BERT model to the attack technique of this ontology. Specifically, K-CTIAA inputs the extracted threat information into the ontology as a query, and then calculates the similarity between each candidate attack technique and threat information according to the algorithm to find out the most likely attack technique. Finally, according to the mapping relationship of this attack technology, we can find the corresponding countermeasures of this threat action in D3defend.

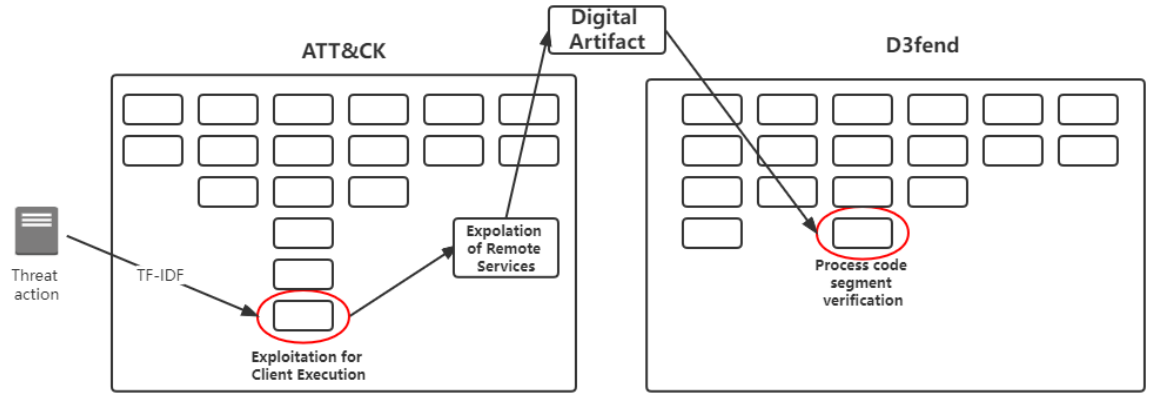

Figure 8. Countermeasure search process.

4. Result

The dataset we use is an open-source APT report repository named APTNotes [35]. The entire repository contains 634 APT reports from different security companies, including different APT groups and different years. We use the PDFMiner (a format conversion tool) to convert the reports in PDF format to text format. In fact, this method can also handle cyber threat intelligence from Twitter or other forms, just by converting the cyber threat intelligence to a text format as required.

In the first experiment, we compared K-CTIAA with some previous cyber threat intelligence analysis tools, which we reproduced and tested on the dataset, including TTPDrill [10], and ActionMiner [13]. These results are all summarized in Table 1. In order to prevent the difference in implementation details from causing the model to be less effective than expected, we have also reproduced some commonly used neural network models, including Convolutional Neural Network (CNN), Bidirectional Gate Recurrent Unit (Bi-GRU), Bidirectional Long Short-Term Memory Network (Bi-LSTM) and BERT [36] model. These results are all summarized in Table 2.

Table 1. Comparison between several cyber threat intelligence analysis tools: TTPDrill, ActionMiner and K-CTIAA.

	TTPDrill	ActionMiner	K-CTIAA
P	0.907	0.923	0.931
R	0.881	0.929	0.951
F1	0.894	0.926	0.941

Table 2. Comparison between different neural network models and K-CTIAA.

	CNN-CRF	BiGRU-CRF	BiLSTM-CRF	BERT-CNN-CRF	BERT-GRU-CRF	BERT-BiLSTM-CRF	K-CTIAA
P	0.813	0.829	0.871	0.865	0.881	0.887	0.931
R	0.862	0.886	0.911	0.890	0.912	0.926	0.951
F1	0.837	0.857	0.891	0.877	0.896	0.906	0.941

Moreover, we verify the effectiveness of our two improvements in knowledge fusion on K-CTIAA through two other experiments. K-CTIAA without sentence tree pruning is added to the experiment as a control group, and this group will be compared with K-CTIAA to verify the effect of sentence tree pruning. Likewise, we design another experiment to verify the performance improvement brought by modifying the visible matrix construction rules. A model that builds rules using the matrix visible on in Figure 4 will be compared with the results of K-CTIAA. All the experimental results are shown in Table 3

and Figure 9; K-CTIAA/WO_STP represents K-CTIAA without sentence tree pruning, and K-CTIAA/WO_MM represents K-CTIAA without visible matrix rule modification.

Table 3. Experimental result of ablation experiment.

	K-BERT	K-CTIAA	K-CTIAA/WO_STP	K-CTIAA/WO_MM
P	0.897	0.931	0.926	0.911
R	0.936	0.951	0.950	0.940
F1	0.916	0.941	0.938	0.925

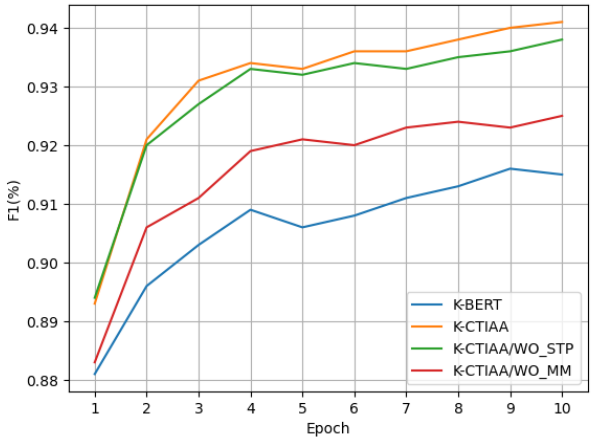

Figure 9. F1 score of ablation experiment.

In the second experiment, we explored the threshold for pruning. First, we compared the F1 scores of K-CTIAA with different thresholds. The optimal solution to this threshold may exist in many cases, but for the sake of our analysis, we put forward an assumption that only one optimal solution will appear in the threshold of control pruning. This change is continuous, and the optimal solution cannot be obtained if the threshold is set higher or lower than this value. Based on this assumption, we set different thresholds to calculate the F1 score of K-CTIAA under the same conditions, and the results are shown in Table 4. Secondly, in order to understand the influence of different thresholds on the number of pruning, we calculated the average number of knowledge branches inserted by each entity under different thresholds, which can help us intuitively understand the influence of threshold setting on the model. The detailed data are shown in Table 5.

Table 4. F1 score of K-CTIAA under different pruning thresholds.

Threshold	0	0.02	0.04	0.06	0.08	0.10
F1 Score	0.937	0.937	0.938	0.940	0.941	0.938

Table 5. Average number of knowledge branches inserted into each entity.

Threshold	Average Number of Knowledge Branches
0	5.4
0.02	4.7
0.04	4.7
0.06	4.6
0.08	4.3
0.10	3.9

Among the experiment results, the F1 score of K-CTIAA is significantly higher than that of the BERT model without knowledge graphs, which confirms that the knowledge

graph can help the pre-trained model to better understand semantic information and thus achieve higher accuracy, while the sentence tree pruning has limited improvement on K-CTIAA. We speculate that there may be several reasons as follows. First of all, the complexity of the network security knowledge graph is relatively simple, which will lead to relatively few entities associated with various knowledge, and the scope of application of pruning rules is small. Secondly, there are certain shortcomings in the correlation algorithm between sentences and knowledge. TF -IDF cannot reflect the correlation between the two well. Relatively speaking, modifying the visible matrix construction rules can also significantly improve the effectiveness of K-CTIAA. This proves that knowledge also needs to be understood in context. By modifying the construction rules of the visible matrix, the semantic information of the context can also be integrated into the knowledge, so that the semantics of the knowledge in different contexts can be more accurate.

5. Discussion

In this section, we first briefly analyze the experimental results, and then discuss simply the following five issues:

- How to set the pruning threshold.
- Why does the visible matrix need to be changed? Does the inserted knowledge need to acquire the semantics of the original sentence?
- Why choose a pre-trained model, and how to replace the BERT model in K-CTIAA?
- How does K-CTIAA cope with different requirements?
- Dynamic knowledge graph completion.

5.1. How to Set the Pruning Threshold

It is clear that the pruning of the sentence tree is necessary and useful, and experimental results have proved this point. Regarding the question of the threshold setting, this paper conducts experiments by setting different thresholds and compares the experimental results to determine the pruning threshold of K-CTIAA. In theory, the number of triples to be kept is closely related to the dataset and knowledge graph. The larger the knowledge graph, the greater the amounts of relevant knowledge it contains and knowledge it needs to retain. At the same time, the larger the scale of the knowledge graph, the higher the frequency of pruning that is required, and the lower the pruning threshold that needs to be set, because the relevance of knowledge will decrease. It is unwise to set the knowledge pruning threshold arbitrarily. As shown in Table 5, there is no obvious boundary value for the division of thresholds, and changing the rules for judging knowledge is also one of the feasible tasks in the future. In addition, different entities will have large gaps in calculating the relevant scores related to knowledge, so there are also theoretical flaws in dealing with different entities according to established standards, which is also the future work of K-CTIAA. We believe that a better method is to learn the filtering process through the model itself based on semantic filtering knowledge, but this learning process requires sufficient data support, and the lack of a large amount of data is also one of the tasks that need to be completed in the field of network threat intelligence analysis.

5.2. Is Knowledge Required to Acquire the Semantics of the Original Sentence?

We believe the answer to this question is yes, and our experimental results confirm this. Knowledge has different meanings in different contexts. Logically speaking, knowledge can also be regarded as a special sentence, and the meaning of sentences also needs to consider the special context in which it is located. In fact, scholars have already undertaken research in this regard. Researchers from the Microsoft team leverage external entity descriptions to provide contextual information for graph entities [37]. Their proposed model achieved SOTA on the Commonsense QA benchmark. For this problem, the measures we take do not require the introduction of additional contextual information, and only use the form of triples for knowledge fusion. By modifying the visible matrix, the knowledge can obtain

the semantic information of the input context. However, on the starting point, our ideas are the same: we all believe that knowledge itself cannot be understood in isolation.

5.3. Choice of the Pre-Trained Model

In the experimental part of this post, we compared K-CTIAA with the BERT model, and we performed better than other methods. However, the current pre-trained model is developing rapidly, and the BERT model has been overtaken by other pre-trained models. Does this mean that our method is backward? We believe that the answer is no. In Section 3, we also explained the advantages of the K-BERT model structure: it can be compatible with other pre-trained models. In fact, we only need to change the basic model of K-CTIAA from BERT to any other pre-trained model based on transform through some simple changes. We only need two steps to transform a pre-trained model into the architecture of the K-CTIAA model: the first step is to add a knowledge query and just inserted modules before the input of the pre-trained model, and transform the input into the input of the model format; the second step is to add the visible matrix M to the calculation formula of self-attention in transform. After completing these two steps, K-CTIAA is able to upgrade its cores, thereby improving its performance. One purpose of this paper is to verify that the knowledge graph can improve the performance of the pre-trained model in the field of cyber threat intelligence analysis, rather than simply pursuing higher F1 scores. Changing the pre-trained model can achieve better performance, but that is not the focus of our research. Another point to consider is that the iteration speed of the current pre-trained model is too fast, and the SOTA model is constantly changing. We believe that the more basic BERT model is more representative.

5.4. How Does K-CTIAA Cope with Different Requirements?

In Section 3, we talked in detail about how a phishing email attack could be matched into a countermeasure, but not all situations require finding a countermeasure. In some cases, users may want to know more about the relevant information of this threat action. For example, users would like to see threat intelligence in a standard sharing format, or extracted Indicators of Compromise organized as a graph, or TTPS of threat actions based on analysis. In fact, these organizational forms are based on threat actions. With a little modification, K-CTIAA can transform the extracted threat actions into the corresponding format. Suppose that if K-CTIAA needs to analyze the TTPS of a threat action, K-CTIAA can first extract the relevant technologies by the threat action, then match them into the corresponding Technique, and then complete the TTPS of this threat action according to the prior knowledge in ATT&CK. Similarly, we can make different processing of the threat actions and Indicators of Compromise extracted by K-CTIAA to meet different requirements.

5.5. Dynamic Knowledge Graph Completion

A knowledge graph is generally constructed manually or learned directly through representation learning. For such an incomplete knowledge graph obtained manually or semi-automatically, especially for sparse or implicit relationships that have not been excavated, if there is a method to complete it to a certain extent, the knowledge graph can become more complete.

At present, we have some ideas and problems about the knowledge graph completion. We have three proposals. The first method is to use the existing knowledge graph to help improve our own. In this paper, the knowledge graph we used is based on the existing knowledge graph. We can track the updates of these knowledge graphs and synchronize their updates at any time through the crawler tool, but this method requires authoritative and constantly updated data sources.

The second method is to use the knowledge graph completion technology to complete the knowledge graph in the process of use. This scheme is similar to the learning process of human beings, which means that knowledge is learned during the process of analyzing threat intelligence, and this knowledge will be added to the knowledge graph. However,

the correctness of the knowledge extracted by this method needs to be carefully considered and may require the participation of experts in the review.

The third method is similar to the second, but we think this one is more reliable. We no longer rely on the model to directly learn knowledge, but allow the model in the training process to recognize entities; this task was better solved in NLP. At the same time, we train a model, through which the relationship between entities is predicted. This method also has the problem of knowledge correctness.

6. Conclusions and Future Work

This paper proposes a network threat intelligence automatic analysis tool, KCTIAA. K-CTIAA enhances the model's understanding of cybersecurity terminology by integrating the cybersecurity knowledge graph into the pre-trained model. K-CTIAA proposes two new improved methods for the knowledge fusion of pre-trained models. In addition, based on the relevant knowledge of the network security knowledge graph, K-CTIAA can find information related to threat actors and provide corresponding threat behavior countermeasures, which will help network security experts deal with various network security issues. Experimental results show that K-CTIAA achieves better results than other methods.

In the future, we will work on improving our method and study how to select appropriate knowledge from knowledge graphs. We firmly believe that screening knowledge is necessary, and some recent research may give us inspiration [38]. In addition, research on how to correctly construct the knowledge graph and dynamically update the knowledge map can also be undertaken in the future. Finally, using knowledge through prompt learning can also be investigated.

Author Contributions: Methodology, Z.-X.L. and Y.-J.L.; software, Z.-X.L.; validation, C.L. and N.-X.Z.; formal analysis, Y.-W.L.; writing—original draft preparation, Z.-X.L.; writing—review and editing, Y.-J.L. and Y.-W.L.; project administration, Y.-J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number No.2019QY1406.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vermesan, O.; Friess, P. (Eds.) *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*; CRC Press: Boca Raton, FL, USA, 2022.
2. Most Recent Cyber Attacks—Past Three Months. Available online: https://www.fortinet.com/resources/cyberglossary/recent-cyber-attacks (accessed on 4 November 2022).
3. Wagner, T.D.; Mahbub, K.; Palomar, E.; Abdallah, A.E. Cyber threat intelligence sharing: Survey and research directions. *Comput. Secur.* **2019**, *87*, 101589. [CrossRef]
4. Han, X.; Zhang, Z.; Ding, N.; Gu, Y.; Liu, X.; Huo, Y.; Qiu, J.; Yao, Y.; Zhang, A.; Zhang, L.; et al. Pre-trained models: Past, present and future. *AI Open* **2021**, *2*, 225–250. [CrossRef]
5. Satvat, K.; Gjomemo, R.; Venkatakrishnan, V.N. EXTRACTOR: Extracting attack behavior from threat reports. In Proceedings of the 2021 IEEE European Symposium on Security and Privacy (EuroS&P), Vienna, Austria, 6–10 September 2021; pp. 598–615. [CrossRef]
6. D3fend—A Knowledge Graph of Cybersecurity Countermeasures. Available online: https://d3fend.mitre.org/ (accessed on 22 January 2023).
7. Sahrom Abu, M.; Rahayu Selamat, S.; Ariffin, A.; Yusof, R. Cyber Threat Intelligence—Issue and Challenges. *Indones. J. Electr. Eng. Comput. Sci.* **2018**, *10*, 371–379. [CrossRef]
8. MITRE | ATT&CK. Available online: https://attack.mitre.org/ (accessed on 11 January 2023).

9. Niakanlahiji, A.; Wei, J.; Chu, B.-T. A Natural Language Processing Based Trend Analysis of Advanced Persistent Threat Techniques. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018. [CrossRef]
10. Husari, G.; Al-Shaer, E.; Ahmed, M.; Chu, B.; Niu, X. TTPDrill. In Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, 4–8 December 2017. [CrossRef]
11. Fujii, S.; Kawaguchi, N.; Shigemoto, T.; Yamauchi, T. CyNER: Information Extraction from Unstructured Text of CTI Sources with Noncontextual IOCs. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2022; pp. 85–104.
12. Noor, U.; Anwar, Z.; Amjad, T.; Choo, K.K.R. A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise. *Future Gener. Comput. Syst.* **2019**, *96*, 227–242. [CrossRef]
13. Husari, G.; Niu, X.; Chu, B.; Al-Shaer, E. Using Entropy and Mutual Information to Extract Threat Actions from Cyber Threat Intelligence. In Proceedings of the 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), Miami, FL, USA, 9–11 November 2018. [CrossRef]
14. Ramnani, R.R.; Shivaram, K.; Sengupta, S. Semi-automated information extraction from unstructured threat advisories. In Proceedings of the 10th Innovations in Software Engineering Conference, Jaipur, India, 5–7 February 2017; pp. 181–187. [CrossRef]
15. Sun, T.; Yang, P.; Li, M.; Liao, S. An Automatic Generation Approach of the Cyber Threat Intelligence Records Based on Multi-Source Information Fusion. *Future Internet* **2021**, *13*, 40. [CrossRef]
16. Preuveneers, D.; Joosen, W. Sharing Machine Learning Models as Indicators of Compromise for Cyber Threat Intelligence. *J. Cybersecur. Priv.* **2021**, *1*, 140–163. [CrossRef]
17. Pujara, J.; Miao, H.; Getoor, L.; Cohen, W. Knowledge Graph Identification. In *International Semantic Web Conference*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 542–557.
18. Herzog, A.; Shahmehri, N.; Duma, C. An ontology of information security. *Int. J. Inf. Secur. Priv.* **2007**, *1*, 1–23. [CrossRef]
19. Jia, Y.; Qi, Y.; Shang, H.; Jiang, R.; Li, A. A practical approach to constructing a knowledge graph for cybersecurity. *Engineering* **2018**, *4*, 53–60. [CrossRef]
20. Syed, Z.; Padia, A.; Finin, T.; Mathews, L.; Joshi, A. UCO: A unified cybersecurity ontology. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*; AAAI Publications: New York, NY, USA, 2016.
21. Noel, S.; Harley, E.; Tam, K.H.; Limiero, M.; Share, M. CyGraph: Graph-based analytics and visualization for cybersecurity. In *Handbook of Statistics*; Elsevier: Amsterdam, The Netherlands, 2016; Volume 35, pp. 117–167. [CrossRef]
22. Ren, Y.; Xiao, Y.; Zhou, Y.; Zhang, Z.; Tian, Z. CSKG4APT: A Cybersecurity Knowledge Graph for Advanced Persistent Threat Organization Attribution. *IEEE Trans. Knowl. Data Eng.* **2022**. [CrossRef]
23. Li, K.; Zhou, H.; Tu, Z.; Feng, B. Cskb: A cyber security knowledge base based on knowledge graph. In Proceedings of the International Conference on Security and Privacy in Digital Economy, Quzhou, China, 30 October–1 November 2020; Springer: Singapore, 2020; pp. 100–113.
24. Wang, P.; Liu, J.; Hou, D.; Zhou, S. A Cybersecurity Knowledge Graph Completion Method Based on Ensemble Learning and Adversarial Training. *Appl. Sci.* **2022**, *12*, 12947. [CrossRef]
25. HoloLen/Cybersecurity_Knowledge_Graph. Available online: https://github.com/HoloLen/CyberSecurity_Knowledge_graph (accessed on 4 November 2022).
26. Aida-yy/Knowledge-Graph-for-Security. Available online: https://github.com/Aida-yy/Knowledge-graph-for-security (accessed on 4 November 2022).
27. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional Transformers for language understanding. *arXiv* **2018**, arXiv:arXiv:1810.04805, 04805.
28. Cao, Y.; Sun, Z.; Li, L.; Mo, W. A Study of Sentiment Analysis Algorithms for Agricultural Product Reviews Based on Improved BERT Model. *Symmetry* **2022**, *14*, 1604. [CrossRef]
29. Lu, Y.; Yang, R.; Jiang, X.; Zhou, D.; Yin, C.; Li, Z. MRE: A Military Relation Extraction Model Based on BiGRU and Multi-Head Attention. *Symmetry* **2021**, *13*, 1742. [CrossRef]
30. Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Ju, Q.; Deng, H.; Wang, P. K-bert: Enabling language representation with knowledge graph. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 2901–2908. [CrossRef]
31. Summary | Three Major Paths, An Overview of the Research Progress of the Knowledge Map Fusion Pre-Training Model. Available online: https://mp.weixin.qq.com/s/9Gw7K1g3u0gPYl48U3WREA (accessed on 4 November 2022).
32. Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; Liu, Q. ERNIE: Enhanced language representation with informative entities. *arXiv* **2019**, arXiv:1905.07129, 2019.
33. Wang, X.; Gao, T.; Zhu, Z.; Zhang, Z.; Liu, Z.; Li, J.; Tang, J. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguist.* **2021**, *9*, 176–194. [CrossRef]
34. PDFMiner. Available online: https://pdfminer-docs.readthedocs.io/pdfminer_index.html (accessed on 4 November 2022).
35. kbandla/APTnotes: Various Public Documents, Whitepapers and Articles about APT Campaigns. Available online: https://github.com/aptnotes/data (accessed on 4 November 2022).
36. Li, Z.; Li, Y.; Zhang, H.; Li, J. Construction of TTPS from APT Reports Using Bert. In Proceedings of the 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 17–19 December 2021; pp. 260–263. [CrossRef]

37. Xu, Y.; Zhu, C.; Xu, R.; Liu, Y.; Zeng, M.; Huang, X. Fusing context into knowledge graph for commonsense reasoning. *arXiv* **2020**, arXiv:2012.04808.
38. Ye, H.; Zhang, N.; Deng, S.; Chen, X.; Chen, H.; Xiong, F.; Chen, X.; Chen, H. Ontology-enhanced Prompt-tuning for Few-shot Learning. In Proceedings of the ACM Web Conference 2022, Lyon, France, 25–29 April 2022; pp. 778–787. [CrossRef]