APTBert: Abstract Generation and Event Extraction from APT Reports

Chenxin Zhou[1], Cheng Huang[1,2(✉)], Yanghao Wang[1], and Zheng Zuo[3]

[1] School of Cyber Science and Engineering, Sichuan University, Chengdu, China
opcodesec@gmail.com
[2] Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation, Hefei, China
[3] Chengdu University of Information Technology, Chengdu, China

Abstract. Due to the rapid development of information technology in this century, APT attacks(Advanced Persistent Threat) occur more frequently. The best way to combat APT is to quickly extract and integrate the roles of the attack events involved in the report from the APT reports that have been released, and to further perceive, analyze and prevent APT for the relevant security professionals. With the above issues in mind, an event extraction model for APT attack is proposed. This model, which is called APTBert, uses targeted text characterization results from the security filed text generated by the APTBert pre-training model to feed into the multi-head self-attention mechanism neural network for training, improving the accuracy of sequence labelling. At the experiment stage, on the basis of 1300 open source APT attack reports from security vendors and forums, we first pre-trained an APTBert pre-training model. We ended up annotating 600 APT reports with event roles, which were used to train the extraction model and evaluate the effect of event extraction. Experiment results show that the proposed method has better performance in training time and F1(77.4%) as compared to traditional extraction methods like BiLSTM.

Keywords: Advanced Persistent Threat · Event Extraction · Abstract Generation · Pre-training

1 Introduction

Since the 21st century, APT incidents emerge in an infinite number of ways, resulting in enormous impact on individuals, organizations and society. In order to better prevent APTs, relevant security practitioners must first be aware of and understand APTs from prior attack events, analyze and summarize their features and laws, so as to better combat APTs [1]. Given that there are a large number of APT reports available on the Internet (official websites of security vendors, forums, social platforms, etc.), this paper aims to automatically extract attack events from APT reports, so that security practitioners can find the core

S. Goel and P. R. Nunes de Souza (Eds.): ICDF2C 2023, LNICST 571, pp. 209–223, 2024.
https://doi.org/10.1007/978-3-031-56583-0_14

content and key actors of the event more quickly. Furthermore, this topic can also provide a role extraction scheme for constructing a knowledge graph of attack events, and then become the basis for reasoning about downstream knowledge. If a new APT event appears on the open source website, the extraction work can be completed immediately and the elements of the event can be added to the event knowledge graph.

By analyzing existing attack event extraction mining technology for cyber security attack reports, in this paper, we find that the text characterization from most of the extraction methods is not effectively represented. With this in mind, the paper investigates how to perform a more comprehensive characterization of the security text. Furthermore, existing work is not complete for the role definition for APT events. To address the aforementioned issues, this paper proposes a set of event role definition and extraction schemes for APT reports, which can efficiently extract key information in APT reports.

In summary, the main innovation points and contributions of the paper are as follows:

- We introduce a TextRank algorithm based on similarity between sentences to extract important sentences from the APT report in order to generate a abstract, which is convenient for quickly understanding the core content of the report.
- We define 14 types of key event roles in APT attacks, and collect 11,082 sentences from 600 APT reports and annotate them as the training and evaluation dataset of the model.
- We collect 863 texts in the security field for self-supervised pre-training. And we generate the APTBert pre-training model, then perform text characterization based on this model. It enables rare texts in the security field to be effectively represented. And we put forward the event role extraction method based on APTBert and conduct experiments, compared with the traditional extraction model, it performs better in the experiment result.

The remainder of this article is organized as follows. **Section** 2 surveys the relevant work about abstract generation and event extraction. Methods of abstract generation and event extraction are described in **Sect.** 3. Our experiment design and implementation and its result and analysis will be seen in **Sect.** 4. Limitations are presented in **Sect.** 5. Finally, we conclude our contributions and improvements in **Sect.** 6.

2 Related Work

2.1 Abstract Generation

Abstract generation methods are mainly divided into two categories: extraction-based method and abstraction-based method [2]. The typical extraction-based method is TextRank algorithm, which is proposed by Mihalcea [3] et al., it generates sentence weights based on the similarity between words and the frequencies

between sentences, and then extracts high-weight sentences for sequential combination so that will be no grammatical problems and the key points of the full text can be grasped. In addition, the typical abstraction-based method is the text topic model LDA [4]. We will only consider whether the next word appears at the same time, without taking into account the order in which the words appear. However, it still requires a lot of extra training time for the encoder and decoder.

2.2 Event Extraction

Event extraction techniques are usually trained with text characterization as the basis for supervised deep learning. In the general field, Malte [5] et al. splice the characterization of the open source knowledge map as the feature input to represent the Bert pre-training model; Dongfang Lou [6] et al., construct a multilayer two-way network model which is used as an encoder and a decoder; Yubo Chen [7] et al. use a convolution neural network model with multiple pooling layer structures to semantic information relations are extracted. In the security field, researchers often combine excellent models in general fields with text features in the security field to propose event extraction schemes that can perform better on security texts. For example, Semih [8] et al. use CNN [9] as an encoder to encode the result of embedding in a new way, and as the input of the CRF layer, it performed well on short texts containing noise; Ning Luo [10] et al. propose a BiLSTM model with a sliding window to fuse inter-sentence information, and finally proved that the text-level learning model performs better than the sentence-level learning model in the role extraction of most event; Jakub [11] et al. use information open source platforms such as Twitter to collect intelligence related to border security; furthermore, some work focuses on introducing external features to improve the recall and precision of the model. For example, Benjamin [12] et al. use alarm graphs and event feature embedding to detect APT attacks. For sequential data like text, a common learning network structure is RNN [13]. However, the recurrent neural network system is not effective in dealing with long text sequences. Then the LSTM was originally designed and the "forget gate" was introduced [14]. However, the model structure of LSTM itself is relatively complex, there is a bottleneck for LSTM for longer sequence data. A multi-headed self-attention neural network performs better in solving sequence tokens [15]. And the multi-headed self-attention mechanism acts as an integration to prevent overfitting. This model can greatly improve the computational speed and learning ability, and facilitate the performance and processing of very long texts. This enables the text to be processed from sentence level to text level so that the accuracy of prediction can be improved.

3 Methodology

In this section, we first study on combining sentence similarity and sentence centrality to generate an abstract through the TextRank algorithm. Then we

present our APTBert pre-training model to extract APT events. The overall design framework of APT report abstract generation and event extraction is shown in Fig. 1. First, the collected dataset is cleaned and then two parallel processes are performed: abstract generation and event extraction. In addition, the abstract generation module, which is mainly relied on TextRank algorithm, takes the whole APT report as input and extract the APT report through APTBert pre-training model.

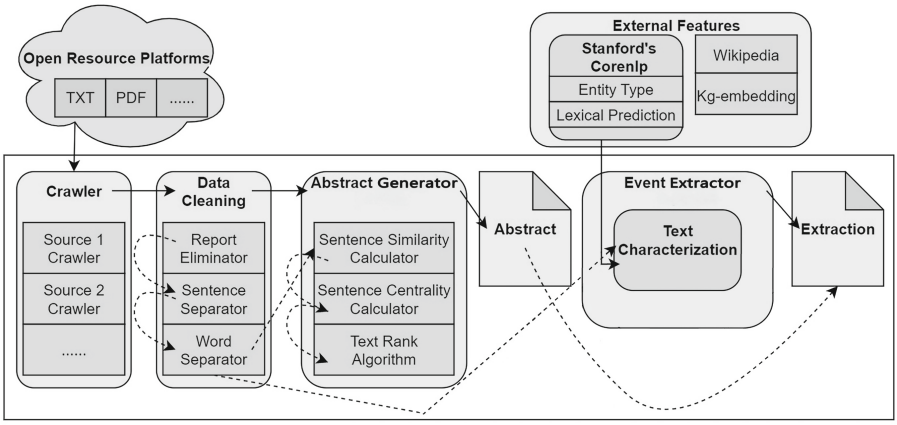

Fig. 1. Overall Framework

3.1 Data Cleaning

This module contains three steps: report elimination, sentence separation, word separation. For report elimination, we eliminate irrelevant or low-quality reports artificially. And then we use scripts, which are programmed in Python, to divide sentences of a report and each sentence is divided into words based on spaces and punctuation marks. Each word and punctuation is treated as a separate token, and the BRAT annotation tool is used to annotate the event roles.

3.2 Abstract Generator

In this section, we present abstract generator method based on TextRank algorithm. First we get word separations from previous stage, and we calculate the sentence similarity according to them. Then we calculate the sentence centrality which is based on the sentence similarity. After that, we set a sentence similarity value as the threshold, if one's sentence similarity is bigger than the threshold, it will be generated as part of the abstract. The TextRank algorithm flow diagram is as follow.

Algorithm 1. TextRank Algorithm

```
INPUT : DocumentT
spilt(T)into[S1, S2, ..., Sm]
spilt(Si)into[t1, t2, ..., tn]
for i in m do
    for j in m do
        Similarity(S_i, S_j) = (sum_{k=1} (t_k ε S_i ∧ t_k ε S_j)) / (log^{|S_i|} + log^{|S_j|})
    end for
    for i in n do
        S_i = sum_{j=1}^{N} Similarity_{i,j} / N
        if Si > threshold then
            Res.append(Si)
            ReturnRes
        end if
    end for
end for
```

Sentence Similarity Calculation. For a given piece of text T, it is first divided into sentences according to its content (according to punctuation splits):

$$T = [S_1, S_2, \ldots, S_m] \tag{1}$$

After the clause is completed, go out and deactivate the word and split the word for each clause:

$$S_i = [t_{i,1}, t_{i,2}, \ldots, t_{i,n}] \tag{2}$$

After each sentence has been divided, the similarity can be calculated based on the co-occurrence frequency between the two sentences, and for a given two sentences:

$$Similarity(S_i, S_j) = \frac{\sum\limits_{k=1} (t_k \epsilon S_i \wedge t_k \epsilon S_j)}{log^{|S_i|} + log^{|S_j|}} \tag{3}$$

A higher similarity score of two sentences indicates a stronger association. Sentence similarity is also the basis for calculating the importance of sentences in the text.

Sentence Centrality Calculation. We introduce sentence centrality based on sentence similarity to describe how the sentence is related to other sentences in the full text and thus can quantitatively represent the importance of the sentence. After getting the similarity between sentences then it can be used as the weight between sentences. Let the weight between sentences i and j be Similarity, then the centrality S of sentence i is as below:

$$S_i = \sum_{j=1}^{N} Similarity_{i,j}/N \tag{4}$$

The centrality of a sentence calculates the arithmetic mean of the degree of association of the sentence with other sentences in the full text. If this mean is higher then it indicates that the sentence is of high importance in the full text, then the choice of whether or not to extract the sentence can be made based on this value, and the larger the centrality value the more likely it is that the sentence will be extracted as part of the abstract.

Abstract Generation. We include a sentence as part of the abstract when its importance score is greater than the set threshold. It allows the more important sentences in the text to be filtered out to generate an abstract. We input longer security reports piece by piece for abstract extraction, then we can have an overall grasp of the general meaning of the text.

3.3 Event Extractor

In this section, we first have to know what is needed to be extracted so that we define 14 types of APT attack event roles. And characterizing information as a tensor is the foundation of deep learning. So we propose a pre-training framework similar to Bert, which is called APTBert. And we output the characterization through multi-headed self-attention mechanism. Finally we extract an event through this method.

Role Definition. Based on previous work related to cyber attack event extraction, we define 14 types of roles in attack events (see Table 1 for specific role types and definitions) for which can be extracted automatically, thus helping relevant practitioners to understand the attack event and work further based on it.

Based on the work of Satyapanich [16] et al. the roles of cyber security events are classified into 13 roles of attacker, victim, attack mode, attack tool, loss, number of victims, purpose, location, time, CVE, vulnerability, vulnerable system, and version of vulnerable system. We added five new event role types of victim location, sensitive information, attacker clues, means of defense and related activities. In addition, we merged CVE vulnerabilities and other vulnerabilities (unified as the vulnerabilities exploited by the attack), merged the number of victims into the victim role as well, and merged the version of the vulnerability system into the vulnerability system. This allows for a more comprehensive division of APT events in terms of roles, thus improving the extraction of events.

Text Characterization. According to APTBert pre-training model, the token embedding is generated by first quantizing the tokens in successive sentences using word2vec. Segment embedding is used to distinguish the context and position embedding characterizes the position of a token in a sentence as a tensor, so that the position of each token can be learned by the model. The position of each token can be learned by the model. The position representation formulas are given in formula (1) and formula (2):

Table 1. Attack event role types and definitions

Event Roles	Definition
Attackers	Entities that initiate cyber attacks against other entities in an APT event
Victims	Entities subject to attacker-initiated cyber attacks in APT incidents
Related Activities	Related activities involved in the APT event
Attack Time	The time when the attacker launched the cyber attack
Victim Location	The country or region where the victim is located
Means of Attack	The technical means utilized by the attacker during the cyber attack
Attack Tools	Tools utilized by attackers in the course of a network attack
Sensitive Information	Important data or information about the victim
Purpose of The Attack	The purpose of cyber attacks carried out by attackers
Damage Caused	Financial and reputational damages caused in the APT incident
Vulnerabilities Exploited By The Attack	Software, hardware, protocol or system vulnerabilities exploited by the attacker in the course of a network attack
Vulnerable Systems	The vulnerable system attacked by the attacker
Defensive Tools	Means of defense against cyber attacks used in the APT incident
Attacker Clues	Information exposed by the attacker in the APT incident

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{5}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{6}$$

The three embeddings are stitched together and fed into an encoder module using a 6-layer self-attention mechanism [17] with multiple heads. Each encoder module finally performs a residual splicing and regularization operation on the tensor. The residual [18] unit can be implemented as a layer-hopping connection, where the input of the unit is directly added to the output of the unit and then activated. For the regularization operation, layer normalization [19] is used in the Bert framework, and BN layer in CNN can accelerate the training of the model and prevent the model from overfitting and gradient disappearance. However, there are some problems if the BN layer is directly applied in RNN [20], if a test text is longer than the training text, Batch Normalization will be problematic. In addition, Stanford provides the already trained natural language processing tool corenlp [21]. It includes named entity recognition, which gives the entity

type of the token. As well as discriminating the lexical nature of the token. We use corenlp's entity type prediction results and lexical prediction results as two external features. In addition, Wikipedia provides node representation based on knowledge graph. Kg-embedding [22] also applies graph representation algorithm for each node. After completing pre-training, the authors stitch the results of characterization via APTBert with the named entity recognition results provided by corenlp, lexical results and features from graph embedding provided by Wikipedia as the final text characterization results to be input to the event role extraction model.

Event Extraction. Unlike the traditional string recurrent neural networks such as LSTM [23] and GRU [24], we use a multi-headed self-attention mechanism for event extraction. This results for better performance in solving sequence tokens with the following equation:

$$Attention\,(K, Q, V) = softmax(QK^t/\sqrt{d})V \tag{7}$$

The multi-headed self-attention mechanism supports parallel token input, making the computation much faster. And the CRF layer allows adding some constraints to the final constraint labels before output, thus ensuring the validity of the prediction labels. In addition, in the loss function of the CRF [25] layer, there are two types of scores: the first score is the emission score, which is calculated by building an emission matrix to obtain the emission score. At the same time, a transfer score matrix is also obtained, which stores the transition score between all tags transferred to each other. Accordingly, this transfer score matrix can be randomly initialized before training the model, and all random scores in this matrix will be updated during the training process, in other words, the CRF layer can learn these constraints by itself without constructing this matrix artificially. The loss function part of the CRF consists of two parts, the score of the true path and the total score of all paths. The score of the true path should be the highest score of all paths. Building each possible path with a score of Pi and a total of N paths, the total score of the paths is

$$P_{total} = P_1 + P_2 + \ldots + P_N = e^{S_1} + e^{S_2} + \ldots + e^{S_N} \tag{8}$$

In turn, the loss function equation is

$$LossFunction = Precision/(P_1 + P_2 + \ldots + P_N) \tag{9}$$

where S is calculated as

$$S_i = EmissionScore + TransitionScore \tag{10}$$

In the output part, the BIO-event role type is used to implement the sequence token. If the prediction result of a token is B, it means that the token is the first token of the role type; if it is I, it is the other part of the role type (except the beginning part); if it is O, it means that the token does not belong to any role type.

Firstly, the token is input, and the named entity recognition and lexical prediction are performed by corenlp to generate the corresponding encoding; in addition, the token is passed through the already trained APTBert pre-training model to generate the token embedding; then the graph features are introduced by kg-embedding. The four features are stitched together and used as the input of the multi-headed self-attention mechanism; after the self-attention mechanism module containing layer normalization and residual linking is input to the CRF layer to learn the constraints and finally generate the predicted probability distribution.

For the prediction results, the paper uses cross entropy as the loss function.

$$L = (\sum_{i} L_i)/N = -\sum i \sum_{c=1}^{M} y_{ic} \log(p_{ic}) \tag{11}$$

Among them: M is number of categories; y_{ic} is symbolic function (0 or 1), if the true category of sample i is equal to c take 1, otherwise take 0; p_{ic} is The predicted probability that observation sample i belongs to category c, and the loss is back-propagated to train the model.

4 Experiments

4.1 Datasets

We crawled through a total of 863 open source APT reports on the official websites of security forums and security vendors (e.g. Kaspersky, HP, IBM), each APT report describes the course of an APT incident. And we used 863 APT reports as the self-training corpus of APTBert. Among the 863 APT reports, 600 reports with 11082 sentences were randomly selected for subscripts, and each word, symbol was considered as a token. Then the event role annotation was performed, and the annotation was decided according to the number of votes. As shown in Table 2, the average number of event role types in each report after annotation was counted.

4.2 Evaluation Metrics

In the event extraction class task, three metrics, precision, recall and F1 value, are usually used to evaluate the effectiveness. The extraction of event elements is divided into two categories from two parts, T stands for the actual value is the target category and F stands for the actual value is not the target category. p stands for the predicted value is the target category and N stands for the predicted value is not the target category. Then TP indicates that both actual and predicted values are target categories. And where the precision is used to measure the probability of making a correct prediction which is calculated as:

$$\text{Precision} = \text{TP}/P \tag{12}$$

Table 2. Statistics on the number of roles in the attack

Event Roles	Amount
Attackers	12.93
Victims	8.66
Related Activities	5.34
Attack Time	2.16
Victim Location	3.72
Means of Attack	3.86
Attack Tools	13.73
Sensitive Information	5.45
Purpose of The Attack	3.56
Damage Caused	2.17
Vulnerabilities Exploited By The Attack	2.08
Vulnerable Systems	3.51
Defensive Tools	4.12
Attacker Clues	0.85

Recall is used to measure the proportion of predictions made that cover all elements, and is calculated as:

$$\text{Recall} = TP/\text{T} \tag{13}$$

Due to the existence of a mutually sacrificial quantitative relationship between precision and recall, the F1 value, which can combine the two, is introduced in order to evaluate the most overall prediction task, where the F1 value is positively correlated with both precision and recall, as calculated by the following formula:

$$F1 = 2 * Recall * Precision/(Recall + Precision) \tag{14}$$

If F1 is higher, it means that the better performance is achieved when for precision and recall importance weights are considered consistent.

4.3 Design and Evaluation

To evaluate the performance of APTBert pre-training model and other methods proposed in this paper, a series of experiments are designed. For our experiments, they can briefly be separated to two parts, abstract generation module and event extraction module, which were presented above in the overall framework.

Abstract Generation Evaluation. The abstract generation module is based on TextRank algorithm. For a piece of APT attack report, after the process of data cleaning, we get the sentence similarity of this report. The mean value of the similarity with all other sentences is used as the score of the importance of the sentence. When the importance score of a sentence exceeds the threshold,

the sentence is extracted as part of the abstract and then we get an abstract of this report. For the result evaluation,we select 50 pieces of APT reports to be generated, and we choose to compare the abstract which is generated by TextRank with the one generated by LDA model. And the average number of sentences of an abstract is also our evaluation criteria to show the performance.

Event Extraction Evaluation. We input the separated words to the APTBert pre-training model. In the self-attention mechanism stage, 6 layers of encoder are set for deep encoding, and each encoder section contains residual linking and regularization operations. The vector characterization is generated through the APTBert model and sent to the multi-headed self-attention mechanism module for feature extraction by stitching with the lexical feature encoding corresponding to the token, the Wikipedia graph node feature encoding and the entity type feature encoding finally through the CRF layer. The probability distribution of each token is learned from the constraints, and the loss value is calculated using cross-entropy as the loss function and then back-propagated to optimize the parameters of the multi-headed self-attention mechanism network. Finally, we compare the result with BiLSTM model to show the performance of APTBert model.

4.4 Result and Analysis

Abstract Generation. The following is the result of an abstract generation for a specific APT report. The abstract can be read to get a general idea of the entire report. The abstract generates the main elements of an APT incident that occurred in October 2018, and it can be seen that the main attack-side players and victim-side players can be presented in the abstract. By generating abstracts for 50 texts using the LDA model and the methods of this thesis pair, the results are as follows:

Table 3. The effect of different methods of abstract generation

Abstract Generation Method	Grammar Error Rate	Average Number of Sentences
LDA	27%	17.6
TextRank	0%	14.8

Compared with other abstract generation methods such as LDA, the results of this experiment have the advantage that grammatical correctness can be guaranteed.

Event Extraction. We divided the 600 reports into 500/50/50 as training set/test set/evaluation set for effect evaluation. The experiments show that using the APTBert pre-training model as the base representation, splicing three external features provided by Corenlp tool, entity type, lexicality and knowledge embedding provided by Wikipedia as input, and trained by multi-headed

self-attention neural network, the performance is better compared with the traditional scheme. The extraction results of all characters were evaluated together and their F1 values were calculated and reached 77.4%. The specific performance is shown in Table 4.

Table 4. Different methods of attack event extraction effect

Embedding	Extraction Model	Recall	Precision	F1
Word2vec	BiLSTM	0.471	0.794	0.591
Glove	BiLSTM	0.509	0.802	0.623
Bert-base-uncased	BiLSTM	0.643	0.886	0.745
APTBert	**Multi-head Self-Attention**	**0.677**	**0.902**	**0.774**

In addition, Table 5 shows a demonstration of the effect of this APT report extraction.

Table 5. Example of event role extraction results

Event Role	Content
Attackers	Cloud Atlas
Victims	industries and governmental entities
Related Activities	cyber-espionage operations
Attack Time	beginning of 2019 until July
Victim Location	Russia, Central Asia and regions of Ukraine
Means of Attack	spear-phishing, spear-phishing emails
Attack Tools	PowerShell backdoor, malicious remote templates, polymorphic VBS implant, polymorphic HTA
Sensitive Information	context file computed by the HTA
Purpose of The Attack	compromise its targets, compromise high value targets, PowerShell and VBS modules to execute on the local computer
Damage Caused	null
Vulnerabilities Exploited By The Attack	CVE-2017-11882, CVE-2018-0802
Vulnerable Systems	Microsoft
Defensive Tools	IoC-based defence
Attacker Clues	null

We can see that the performance of the APTBert-based attack event extraction model in regards to the task of extracting the event elements. Among the

extracted attacker entities are the hacker group Cloud Atlas, the victims are industrial entities and governments, etc., the time of the attack is from July 2019 to launch the attack, using attacks such as phishing emails, and the relevant vulnerabilities exploited are CVE-2017-11882 and CVE-2018-0802.

5 Limitations

The existing problems and possible improvement directions of the paper include the fact that the proposed method of the paper is higher than the traditional methods in terms of recall rate, but still only 67.7%. In order to improve the recall rate, the APTBert pre-training model can be pre-trained on a richer corpus, and the training task of whether it is a contextual sentence can be introduced. In addition, the definition of roles in APT attacks is not comprehensive enough, and more related work can be read to classify the roles of APT events in a more fine-grained way; finally, more external features such as (dependency graphs in corenlp, etc.) can be introduced to do ablation experiments to improve the extraction effect and evaluate the model's relative effectiveness.

6 Conclusion

In this paper, we first investigate and propose an APTBert pre-training model to address the problem that existing work cannot effectively characterize the proprietary vocabulary in the security field. And then we define 14 types of key roles in APT events and identifies two main goals of the APT report-oriented attack event extraction task, abstract generation and event extraction. In addition, we investigate how to generate abstracts from APT reports. And the role of attack events are extracted from APT reports. Finally, in this paper, 863 texts from the security domain were collected for self-supervised pre-training. The APTBert pre-training model is generated and the text is characterized based on this model. It enables the effective characterization of the raw texts in the dedicated domain. And the APTBert-based event extraction method is proposed and experimented, which has better performance in the experimental results compared with the traditional extraction model, reaching 77.4% of F1 value.

Acknowledgment. This work was supported in part by National Key Research and Development Program of China (No.2021YFB3100500), Sichuan Science and Technology Program (No.2023YFG0162), and Open Fund of Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation (No.CSSAE-2021-001).

References

1. Moon, D., Im, H., Kim, I., et al.: DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing APT attacks. J. Supercomput. **73**, 2881–2895 (2017). https://doi.org/10.1007/s11227-015-1604-8

2. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Conference on Empirical Methods in Natural Language Processing (2015)
3. Mihalcea, R., Tarau, P.: TextRank: bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 404–411 (2004)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**(Jan), 993–1022 (2003)
5. Ostendorff, M., et al.: Enriching BERT with knowledge graph embeddings for document classification. arXiv preprint: arXiv:1909.08402 (2019)
6. Lou, D., et al.: MLBiNet: a cross-sentence collective event detection network. arXiv preprint: arXiv:2105.09458 (2021)
7. Chen, Y., et al.: Event extraction via dynamic multi-pooling convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 167–176 (2015)
8. Yagcioglu, S., et al.: Detecting cybersecurity events from noisy short text. arXiv preprint: arXiv:1904.05054 (2019)
9. LeCun, Y., Bottou, L., Bengio, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
10. Luo, N., et al.: A framework for document-level cybersecurity event extraction from open source data. In: 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 422–427. IEEE (2021)
11. Piskorski, J., Tanev, H., Balahur, A.: Exploiting twitter for border security-related intelligence gathering. In: 2013 European Intelligence and Security Informatics Conference, pp. 239–246. IEEE (2013)
12. Burr, B., et al.: On the detection of persistent attacks using alert graphs and event feature embeddings. In: NOMS 2020–2020 IEEE/IFIP Network Operations and Management Symposium, pp. 1–4. IEEE (2020)
13. Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 300–309. Association for Computational Linguistics, San Diego, California (2016)
14. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000). https://doi.org/10.1162/089976600300015015
15. Guo, Q., Huang, J., Xiong, N., Wang, P.: MS-pointer network: abstractive text summary based on multi-head self-attention. IEEE Access **7**, 138603–138613 (2019). https://doi.org/10.1109/ACCESS.2019.2941964
16. Satyapanich, T., Ferraro, F., Finin, T.: CASIE: extracting cybersecurity event information from text. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 05, pp. 8749–8757 (2020)
17. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
18. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
19. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint: arXiv:1607.06450 (2016)
20. Santurkar, S., et al.: How does batch normalization help optimization? In: Advances in Neural Information Processing Systems, vol. 31 (2018)

21. Manning, C.D., et al.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60 (2014)
22. Wang, Q., Mao, Z., Wang, B., et al.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
24. Chung, J., et al.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint: arXiv:1412.3555 (2014)
25. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint: arXiv:1508.01991 (2015)