BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 14/15 pp531–546
DOI: 10.26599/BDMA.2023.9020032
Volume 7, Number 2, June 2024

Unstructured Big Data Threat Intelligence Parallel Mining Algorithm

Zhihua Li, Xinye Yu, Tao Wei, and Junhao Qian*

Abstract: To efficiently mine threat intelligence from the vast array of open-source cybersecurity analysis reports on the web, we have developed the Parallel Deep Forest-based Multi-Label Classification (PDFMLC) algorithm. Initially, open-source cybersecurity analysis reports are collected and converted into a standardized text format. Subsequently, five tactics category labels are annotated, creating a multi-label dataset for tactics classification. Addressing the limitations of low execution efficiency and scalability in the sequential deep forest algorithm, our PDFMLC algorithm employs broadcast variables and the Lempel-Ziv-Welch (LZW) algorithm, significantly enhancing its acceleration ratio. Furthermore, our proposed PDFMLC algorithm incorporates label mutual information from the established dataset as input features. This captures latent label associations, significantly improving classification accuracy. Finally, we present the PDFMLC-based Threat Intelligence Mining (PDFMLC-TIM) method. Experimental results demonstrate that the PDFMLC algorithm exhibits exceptional node scalability and execution efficiency. Simultaneously, the PDFMLC-TIM method proficiently conducts text classification on cybersecurity analysis reports, extracting tactics entities to construct comprehensive threat intelligence. As a result, successfully formatted STIX2.1 threat intelligence is established.

Key words: unstructured big data mining; parallel deep forest; multi-label classification algorithm; threat intelligence

1 Introduction

Cyber security analysis reports, as a form of internet-based open-source big data, contain extensive descriptions of Tactics, Techniques, and Procedures (TTPs) employed in various network attacks. These TTPs not only derive the behavior characteristics of attackers, but also reveal their malicious intent. Typically, cyber security analysis reports are published in an unstructured or semi-structured format using natural language descriptions. However, natural language cannot be directly understood and processed by machines, necessitating entity extraction of TTPs from the reports, mining relationships between TTPs entities, and transforming them into standard format threat intelligence. On the one hand, Sun et al.[1] made a comprehensive review of the latest research work on Cybersecurity Threat Intelligence (CTI) mining from multiple data sources. They made detailed investigations and reports on threat intelligence mining and threat intelligence sharing. On the other hand, Arıkan and Acar[2] constructed a threat intelligence generation system based on data mining. However, the mining efficiency of this system cannot meet the efficiency requirements and the generated threat

• Zhihua Li, Xinye Yu, and Tao Wei are with School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214122, China. E-mail: zhli@jiangnan.edu.cn; yuxinyeprivate @163.com; 2407117028@qq.com.
• Junhao Qian is with School of IoT Engineering, Jiangnan University, Wuxi 214122, China. E-mail: qjhao@jiangnan. edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2023-08-09; revised: 2023-10-23; accepted: 2023-11-02

intelligence is not in standard format, which is unfavorable for information exchange and sharing. The exploration of TTPs in the field of cybersecurity threat intelligence has long captured people's attention. Husari et al.[3] proposed an automated approach to extract TTPs from unstructured threat intelligence texts. Meanwhile, Ge and Wang[4] devised a method to predict TTPs labels from threat intelligence. Both methods aim to directly extract genuine TTPs from semi-structured threat intelligence texts. Facing the massive volume of internet-based cyber security analysis reports, traditional manual processing approaches have become impractical. Thus, there is an urgent need for a scheme and manner capable of efficiently and automatically extracting TTPs entities and their relationships from large-scale cyber security analysis reports.

In general, of the three, tactics, techniques, and procedures, tactics play the most significant guiding role in the field of cyber security. Specifically, for a given cyber attack event, once the tactics are determined, the corresponding techniques and procedures possess clear directionality. For instance, with respect of the ATT&CK model[5], it includes a total of 14 attack tactics, 224 attack techniques, and 546 attack procedures. For example, there are only 9 attack techniques corresponding to the data theft tactic, including automatic theft and periodic transfer, and among these 9 techniques, there are merely 8 attack procedures, such as traffic duplication and Bluetooth leakage. It can be observed that in the process of big data mining of cyber security analysis reports, TTPs serve as crucial clues. Among them, the primary discovery and successful extraction of attack tactic entities are of utmost key factors, as once the attack tactic entities are successfully identified, it becomes much easier to explore the "entity-relationship-entity" triplets along the "tactics, techniques, and procedures" way. This way efficiently facilitates building threat intelligence. Hereby, this study focuses on the optimization objective of parallel mining of threat intelligence in large-scale cyber security analysis reports published on the Webs, with the specific goal of efficiently extracting tactics entities from these published reports. Addressing on the challenge of mining unstructured data, the following steps are undertaken, first, an automated web crawling process is employed to collect internet-sourced cyber security analysis reports, and tactics category labels are annotated to build a dataset of threat intelligence with tactics categorization. Second, in response to the limitations of low execution efficiency and lack of scalability in the sequential deep forest algorithms, a Parallel Deep Forest-based Multi-Label Classification (PDFMLC) algorithm is proposed. This algorithm leverages broadcast variable strategies and Lempel-Ziv-Welch (LZW) algorithm[6] for table compression to decrease network latency and improve the execution efficiency. Finally, the study further developes and proposes the PDFMLC-based Threat Intelligence Mining (PDFMLC-TIM) method, which utilizes the proposed PDFMLC algorithm for efficiency text classification to cyber security analysis reports. This approach enables high-dimensional and multi-label unstructured data mining, facilitating the acquisition of threat intelligence tactics entities and automatically transforming them into the STIX2.1-formatting threat intelligence. The experiment results demonstrate that the PDFMLC algorithm exhibits favorable node scalability and acceleration ratio. Moreover, the PDFMLC-TIM method efficiently performs text classification on cyber security analysis reports, allowing for effective mining of high-dimensional and multi-label unstructured data, as well as the extraction of information regarding threat intelligence tactics entities. These entities are then automatically transformed into STIX2.1-formattimg threat intelligences.

2 Establishment of Intel-Tactics Dataset

Currently, there is a lack of standardized dataset specifically focusing on threat intelligence within the field of cybersecurity[7]. To bridge this gap, this paper undertakes a comprehensive initiative. We randomly collect and scrape more than 12 900 online open-source cybersecurity analysis reports from sources such as cybersecurity companies. Using our bespoke software for annotation, we create a multi-label tactics dataset for threat intelligence, named "Intel-tactics". The establishment process of Intel-tactics is delineated in Fig. 1. For experimental convenience, Intel-tactics have been categorized into six tactical classifications aligned with the ATT&CK model, as outlined in Table 1. Intel-tactics stands as the primary resource for both training and evaluating the effectiveness of the proposed algorithms.

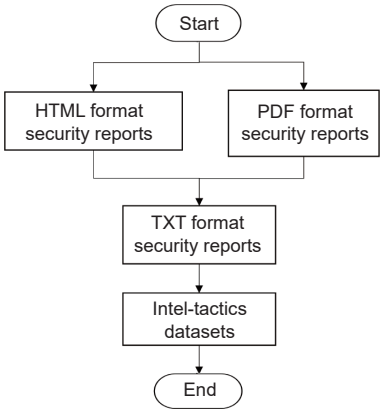


Fig. 1 Process of building Intel-tactics dataset.

3 Parallel Deep Forest Based Multi-Label Data Classification Scheme

The deep forest algorithm exhibits strong learning capabilities to achieve efficient and comprehensive feature mining. This manuscript first investigates and proposes the parallel multi-granularity scanning forest algorithm and the parallel cascading forest algorithm.

3.1 Parallel multi-granularity scanning forest

As shown in Fig. 2, the Multi-Grained Scan (MGS) forest is separated into two parts: sliding window scanning and Spark parallel training. These two parts are shown on the left and right sides of Fig. 2. The sliding window scans each training sample to yield multiple instances, which are then delivered to Spark worker nodes. Multiple Spark worker nodes train random forests in parallel and generate the output classification vectors, which are continuously merged to serve as the input to the cascading forest. The sliding window scanning targets to expand the dimension of input features, creating a richer feature set to enhance the deep forest's learning capability of feature details and improve the algorithm's classification generalization ability. Assuming each input instance has m-dimension features, using a sliding window of size w and moving s dimensions each time, the scanning process will yield new features with dimensions of $w-s+1$. For example, for the standard multi-label dataset "yeast", we set the feature dimension m to 103, the window size to 30, and sliding it by 10 dimensions each time, which results in 8 feature vectors of dimension 30.

After completing the sliding window scanning, the training data are fed to multiple Spark worker nodes. By simultaneously training random forests on multiple Spark worker nodes, parallel training is performed. Herein each Spark worker node independently trains a random forest using the received data and outputs a classification vector. After all Spark worker nodes have completed their training tasks, the classification vectors are sent to the mast Spark node, merging them to produce the output of the multi-grained scan forest.

Although sliding window scanning can yield many features and positively increase the feature dimension of the data instance, it increases additional communication overhead and latency in transmitting the training data to the distributed Spark worker nodes, negatively impacting the overall algorithm's efficiency. To overcome this limitation, inspired by literature[8], the LZW compression algorithm is employed to compress the training data, degrading network transmission data volume and mitigating network transmission delays. When the compressed training data are delivered to the Spark worker nodes, they are then decoded using the LZW algorithm to obtain the original training data. The workflow of the LZW algorithm is illustrated in Fig. 3.

Further, the working mechanism in LZW algorithm is described as below. First, an encoding dictionary table is initialized, containing all possible single

Table 1 Intel-tactics dataset information.

Tactic code	Tactic	Description
TA0002	Execution	Utilizing technical means to execute malicious code on local or remote systems, enabling control over the target.
TA0040	Impact	Attackers attempt to manipulate, disrupt, or compromise systems and data.
TA0005	Defense evasion	Defense evasion refers to the actions taken by attackers to avoid detection throughout the entire intrusion process.
TA0010	Exfiltration	Attackers attempt to steal and disclose data.
TA0011	Command and control	Attackers attempt to communicate with infected machines and send instructions to these devices.
TA0009	Collection	Attackers collect information, and the sources of this information are usually related to the attacker's objectives.

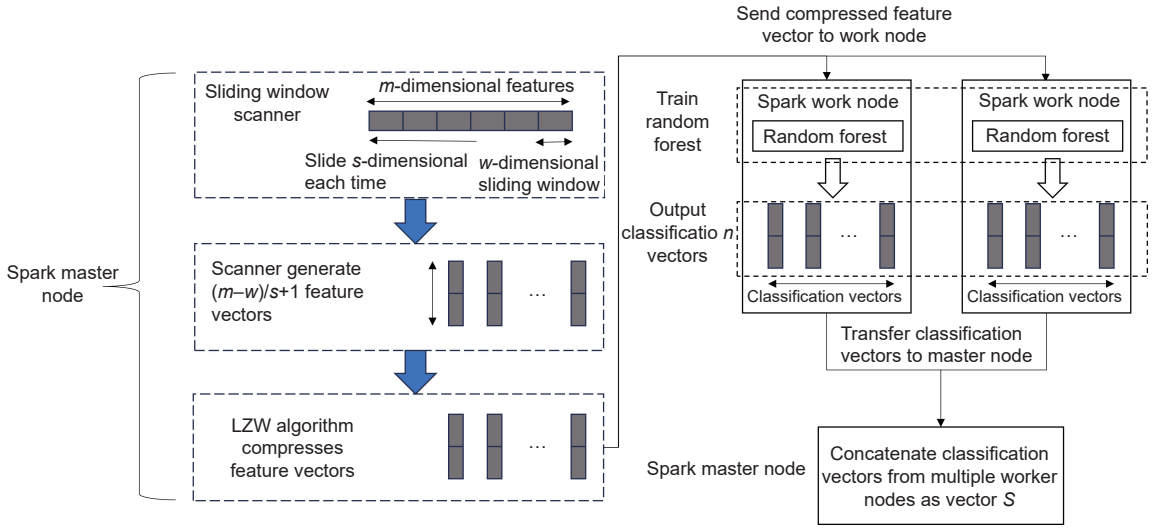


Fig. 2 General view of parallel multi-granularity scanning forest.

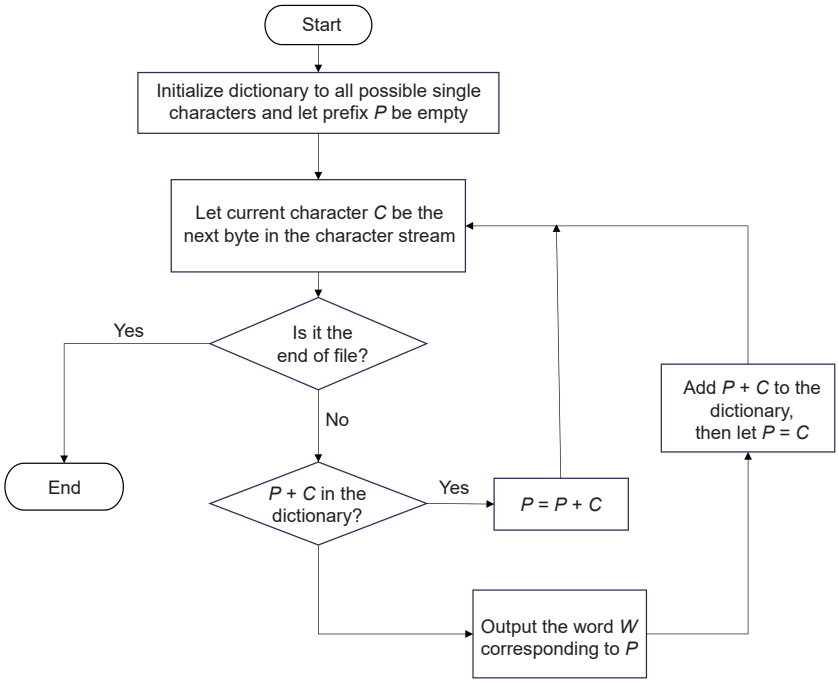


Fig. 3 LZW algorithm process.

characters. Then, the prefix P is initialized as empty, and the next character C in the character stream is read. If the character C is an end-of-file marker, it indicates that all data compression is complete. If the character C is not an end-of-file marker, it checks whether the combination of prefix P and character C exists in the dictionary. If $P + C$ is in the dictionary, the next character in the character stream is read and assigned to C. If $P + C$ is not in the dictionary, the code word W for the prefix P is output, $P + C$ is added to the dictionary, C's value is assigned to P, and then the next character in the character stream is read until the end of the file is reached.

3.1.1 Cascading forest structure

Typically, as the number of layers in a random forest increase, the model's generalization capability gets stronger, allowing it to learn more effective information. This paper proposes the structure of a Cascading Forest (CF), which stacks multiple layers of random forests in a cascading scheme, as shown in Fig. 4.

The input of the cascading forest consists of three

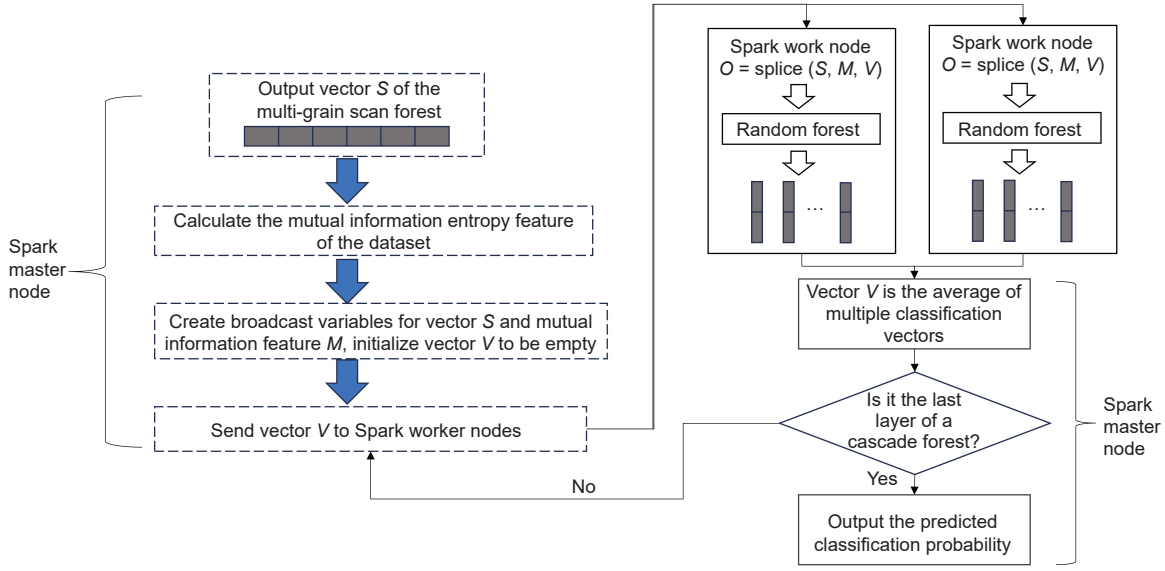


Fig. 4 General view of parallel cascade forest.

parts: the output vector S from the multi-grained scan forest, the label mutual information M of labels in dataset, and the output classification vector V from the previous layer of the CF. When constructing the first layer of the CF, the multi-label mutual information M needs to be computed. The vectors S and the feature M are created as broadcast variables and delivered to the distributed Spark worker nodes. At the Spark worker nodes, the vectors S, multi-label mutual information M, and classification vector V are integrated as the training data for training the random forest. After all the Spark worker nodes in the current layer complete the training tasks, the classification vectors yielded by each Spark worker node are transmitted to the master node, and the average of these classification vectors is calculated to obtain the new vector V. For the second layer of the CF, the average of the classification vectors from the first layer is assigned to vector V, then the random forest is trained accordingly. This process is ongoing until reaching the last layer of the CF.

At this stage, the average of the classification vectors is calculated, which serves as the recognition probability for the target classification labels of the dataset.

3.1.2 Re-usage of broadcast variables

In the parallel CF, the output vector S and label mutual information M from the multi-grained scan forest need to be re-used on the Spark worker nodes, which is shown in Fig. 4. Each level of the CF requires the concatenation of vector S and feature M to obtain the final training data. Namely, vector S and feature M need to be repeatedly used on different levels and Spark nodes to avoid redundant access from the master Spark node. To improve the algorithm's efficiency, reduce the execution time, and avoid data duplication during transmission, the strategy of re-usage broadcast variables is adopted. As illustrated in Fig. 5, vector S and feature M are configured as Spark broadcast variables. Spark broadcast variables are distributed and read-only variables that are stored in the memory of the Spark worker nodes. They can be repeatedly accessed, eliminating the need for redundant data transmission between the Spark nodes.

3.1.3 Label mutual information

Through the analysis of the collected 12 900 cyber security analysis reports, we observe that due to the logical coherence and semantic relationships present in natural language descriptions, there exists a wide range of associations among tactics entities, which often provide detailed descriptions of the emerged attack events.

The capability to discover and explore these associations positively impacts the algorithm's final classification accuracy and practicality. To enhance the sensitivity of the deep forest algorithm to these associations, the label mutual information is cast into the parallel cascade forest and integrated as a feature into the deep forest's input. The label mutual information is represented by

$$H(x) = -\sum p(x) \log (p(x)) \tag{1}$$

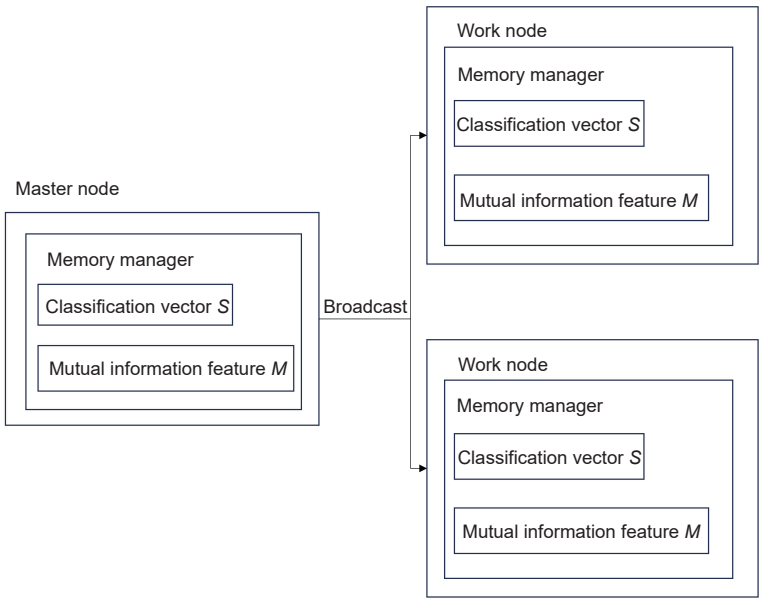


Fig. 5 Multiplexing broadcast variable.

$$H(X|Y) = -\sum_{x,y} p(x, y) \log(p(x \mid y)) \tag{2}$$

where X and Y are the classification labels in the dataset, and x and y are the values of labels X and Y, which can be either 0 or 1, indicating membership or non-membership to the corresponding classification label. According to the Ref. [9], $H(x)$ represents the expectation of the amount of information that is entropy. $H(X \mid Y)$ is defined as the mathematical expectation of the entropy of the conditional probability distribution X over Y under the given condition Y. $p(x)$ represents the joint probability density of values x and y, $p(x)$ and $p(y)$ are the probabilities of values x and y for labels X and Y, respectively. $p(x \mid y)$ is the probabilities of values x which under condition y, $p(x, y)$ is joint density function of x and y. The definition and the calculation of mutal information are as follows:

$$I(X;Y) = H(X) - H(X|Y) \tag{3}$$

$$I(X;Y) = \sum_{x}\sum_{y} p(x, y) \log\left(\frac{p(x, y)}{p(x)\, p(y)}\right) \tag{4}$$

where $I(X;Y)$ denotes the mutual information between labels X and Y. Generally, the higher the mutual information value, the stronger the association between the two labels[10].

In the parallel cascade forest, the mutual information between all labels is computed to select all mutual information which $I(X;Y) > t$, to build the feature vector of mutual information M. The threshold value t is determined through empirical experiment from numerous trials. Subsequently the mutual information feature vector M, the classification vector of the cascade forest V, and the classification vector of the multi-grained scan forest S are combined to create the input features for the next layer of the cascade forest.

3.2 PDFMLC algorithm

With the combination of the proposed parallel multi-grained scan forest and parallel cascade forest, further, we present PDFMLC algorithm, as described in Algorithm 1. In this algorithm, Lines 1–9 represent the parallel multi-grained scan forest, while Lines 10–21 denote the parallel cascade forest. In the proposed PDFMLC algorithm, F represents the input features of the multi-label classification dataset, d denotes the number of layers in the deep forest, DI is the index used for partitioning the dataset into training and test sets, and y represents the algorithm's final predicted label category. The execution process of the algorithm is briefly described as follows. First, the input features F are scanned using a sliding window, and the scan results are delivered to the distributed Spark worker nodes after compression using the LZW algorithm. Then, the dataset index DI is read to create a delivered resilient delivered dataset $\mathrm{RDD}_{\mathrm{DI}}$. Parallel training of random forests is conducted on the distributed Spark work nodes, and the corresponding classification vectors are output. After all Spark worker nodes

Algorithm 1 PDFMLC algorithm

Input: F, d, DI
Output: y //Result of multilabel classification by PDFMLC
1 Calculate Sliding Window Feature (SWF);
2 Use LZW to compress data LZWD;
3 Send data to Spark worker node D to broadcast LZWD;
4 **while** sc.parallelize(DI).map working **do** // Train random forests in parallel by multiple Spark nodes
5 Load data (D, DI [i]);
6 Create dataset RDD_{DI};
7 Train forests on Spark worker node;
8 Calculate class vector S [i];
9 **end**
10 Combine class vector S;
11 Calculate mutual info entropy M;
12 Create broadcast variable S_b;
13 **for** L =1, 2, …, d **do**
14 **while** sc.parallelize (DI).map working **do**
15 Load data to train node and validate node;
16 Train forests on Spark worker node;
17 Calculate class vector V [i];
18 **end**
19 Combain vector V [i] to V;
20 **end**
21 y =V;

complete training task, the classification vectors are transferred to the master node for merging, resulting in the multi-grained scan forest's classification vector S.

Upon completing the multi-grained scan forest stage, the mutual information M for the labels is computed, and both M and the classification vector S are broadcasted to the distributed Spark worker nodes. Next, the label mutual information M, classification vector S, and the classification vector V from the previous layer of the cascade forest are concatenated as inputs to build the cascade forest layer by layer. After training each layer of the cascade forest, the classification vector V is updated, where the update strategy involves taking the average of multiple classification vectors. Finally, the last layer of the cascade forest outputs the predicted label probability.

Regarding the time complexity of the PDFMLC algorithm, it mainly stems from the multi-grained scan forest and the cascade forest. The time complexity of the multi-grained scan forest primarily depends on the process of building the forest model. Let us denote the number of instances as n, the feature dimension as m, and the number of trees in the forest as L. Then, the time complexity of the multi-grained scan is $O(mn\log_2(n))$. On the other hand, the time complexity of the cascade forest mainly arises from constructing the multi-layered random forests. Assuming the cascade forest has K layers, with n instances, m feature dimensions, and L trees in the forest, the time complexity of the cascade forest is $O(Kmn\log_2(n))$. Consequently, the overall time complexity of the PDFMLC algorithm is $O(mn\log_2(n))$. It is worth noting that the PDFMLC algorithm leverages the parallel training of forests on the distributed Spark cluster. By placing these forests on different Spark nodes and training them simultaneously, unnecessary waiting in the sequence mode is effectively avoided, positively decrease the algorithm's execution time.

4 PDFMLC-Based Threat Intelligence Mining Method

To achieve the extraction and classification of tactics entity from cyber security analysis reports, we further propose the threat intelligence mining method based on the PDFMLC (PDFMLC-TIM) method as described in Algorithm 2. The PDFMLC-TIM algorithm is primarily designed for mining PDF or HTML formatted cyber security analysis reports. HTML files are handled using the Beautiful Soup library[11], where irrelevant information, like advertisements, images, and URL links, are filtered out. For PDF-formatted reports, PDFminer[12] is treated for conversion, filtering out unrelated content, such as images and headers. Subsequently, both HTML and PDF formatted reports are unified and converted into text format. The text-formatting data are preprocessed using Term Frequency-Inverse Document Frequency (TFIDF)[13, 14] and encoded into fixed-length output vectors to represent the textual features of the cyber security analysis reports. The TFIDF features serve as the input to the deep forest classification algorithm, which eventually outputs labels for threat intelligence tactics entities, thus achieving automatic classification and recognition of tactics entity. Further, the PDFMLC-TIM method can yield threat intelligence in terms of the STIX2.1 standard based on the extracted tactics entities. The time complexity of the PDFMLC-TIM method mainly comes from the iterative invocation of the PDFMLC algorithm, as shown in Line 11 in Algorithm 2. For each text-format file, the PDFMLC algorithm is called to extract tactics entity. Assuming

Algorithm 2 PDFMLC-TIM threat intelligence parallel mining method

Input: report // Cybersecurity analysis reports
Output: STIXOutput // Threat intelligence in STIX format
1 for t = 1, 2, …, d **do**
2 **if** report is HTML **then** // Filter distractions such as ads to convert HTML to text format
3 Convert HTML to TXT;
4 **else**
5 Convert PDF to TXT;
6 **end**
7 txts [i++];
8 end
9 **for** txt ∈ txts **do**// "txt" is the text that is yet to be mined, and "txts" is the set for "txt"
10 Remove stop words and calculate TFIDF text features for TXT text;
11 Use PDFMLC to extract tactic;
12 Yield threat intelligence in STIX format;
13 STIXOutput [j++]; // Standardize threat intelligence
14 end
15 Combine threat intelligence in STIXOutput based on similarity;
16 Output STIX format threat intelligence

there are p text-format files, the depth of trees in the forest is k, and the number of trees in the forest is q, thus the time complexity is $O(kpq)$.

5 Experiment and Analysis

5.1 Experiment arrangements

The experiment configuration parameters are listed in Table 2. A Spark cluster is set up using 4 servers, i.e., 4 Spark nodes. Each node is equipped with an Intel Xeon 2.20 GHZ processor, 4 GB RAM, and runs on CentOS 7.6.1810 operation system. The Spark cluster environment version used is Spark 3.1.4, and Python version is 3.6.8. The relevant experiment configuration parameters are shown in Table 3. The maximum depth of the parallel deep forest is set to 10, and the cascade forest consists of 8 forest ensembles. The number of parallel nodes is set as 5, and the evaluation indice for decision tree is the Gini coefficient.

Table 2 Experiment configuration.

Parameter	Value
Operating system	CentOS 7.6.1810
CPU	Intel Xeon 2.20 GHZ
Memory/GB	4 GB
Program language	Python 3.6.8
Spark cluster	Spark 3.1.4
Development tool	PyCharm Community Edition 20 202.1

Table 3 Experiment parameters of PDFMLC.

Parameter	Value
max_layer	10
num_forest	8
num_node	5
Criterion	Gini

5.2 Datasets

To validate the effectiveness of the proposed PDFMLC algorithm, four benchmark Mulan multi-label standard datasets are selected[15]. 80% of the instances are randomly extracted without replacement as the training set, while the remaining instances are used as the test set. The Intel-tactics created in Section 2 is primarily used to validate the practicality of the proposed PDFMLC-TIM method for mining the tactics entities of threat intelligence. The detailed information of the experimental datasets is listed in Table 4.

5.3 Evaluation indices

The accelerate ratio is used to express the performance improvement[16] of parallel computing and sequence execution, and the speedup ratio is defined as

$$S_c = \frac{T_1}{T_c} \tag{5}$$

where T_1 represents the time required for sequential execution of the algorithm, T_c represents the time required for running the algorithm on an c-node Spark cluster[17], S_c denotes the accelerate ratio of the algorithm. A larger S_c indicates less time cost in parallel computing mode, thus indicating better performance of the parallel algorithm. When $S_c = c$, S_c is referred to as the desire accelerate ratio, which means excellent scalability of the parallel algorithm.

5.4 Effectiveness of PDFMLC-TIM

PDFMLC algorithm's effectiveness is validated on the

Table 4 Dataset information.

Dataset	Training dataset	Test dataset	Number of features	Number of classes
yeast	1933	484	103	14
scene	1925	482	294	6
enron	1361	341	1001	53
genbase	529	133	1186	27
intel-tactics	10 320	2580	500	5

standard multi-label dataset Mulan. At the same time, we compare it with the algorithms deep forest[18], random forest[19], and extra trees[20]. The experiment results are shown in Fig. 6.

As shown in Fig. 6, the accuracy of PDFMLC on the dataset yeast is 67.39%, which is the best one among the four compared algorithms. On dataset scene, the PDFMLC algorithm achieves an accuracy of 93.24%, outperforming deep forest by 0.48%. Both the algorithms exhibite comparison recognition performance, which can be attributed to the relatively limited number of label categories in the scene dataset and an abundant instance size, resulting in higher classification accuracy for deep forest.

On enron dataset, the PDFMLC algorithm attains an accuracy of 70.24%, whereas deep forest achievs an accuracy of 68.38%, indicating a 1.86% lower accuracy for deep forest in comparison to PDFMLC. For the genbase dataset, the PDFMLC algorithm obtains an accuracy of 86.83%, surpassing deep forest by 0.15%. Across all four datasets, the PDFMLC algorithm consistently outperforms deep forest, particularly demonstrating 1.86% higher accuracy on the enron dataset. This can be attributed to the PDFMLC algorithm's integration of label mutual information, capturing correlations among label categories. Furthermore, by incorporating the original features as new training features and passing them to subsequent cascaded layers for classification, the PDFMLC algorithm effectively improves the classification accuracy. This method contributes to the enhanced performance of the algorithm.

To validate the parallelization effectiveness of the proposed algorithm, the PDFMLC algorithm and the deep forest algorithm are compared on four standard datasets with respect of execution time and speedup ratio, as shown in Figs. 7 and 8, respectively. Deep forest adopts the typical sequence mode, which multiple forests in the same layer are trained serially. PDFMLC algorithm, on the other hand, is a parallelized one of the deep forest algorithm based on Spark cluster, where random forests are delivered to the distributed Spark nodes and trained in parallel computing. The PDFMLC algorithm exhibites the best speedup ratio on the yeast dataset, reaching 4.65. This can be attributed to the fact that the yeast dataset has the largest number of instances among the four compared datasets, with 1933 samples, and its number of classification labels is 2.3 times that of the scene dataset. Due to the smaller instance sizes of the scene and enron datasets, the speedup ratios on these datasets are slightly lower at 4.49 and 4.35, respectively, with a decrease of 0.16 and 0.3 compared to the yeast dataset. The genbase dataset, which has the smallest instance size among the four compared datasets, also achieves a speedup ratio of 4.13. These experiment results demonstrate the potential of the PDFMLC algorithm based on Spark cluster parallelization in handling big data, particularly in scenarios with a large number of instances and classification labels.

Figures 9 and 10 illustrate the comparison of the

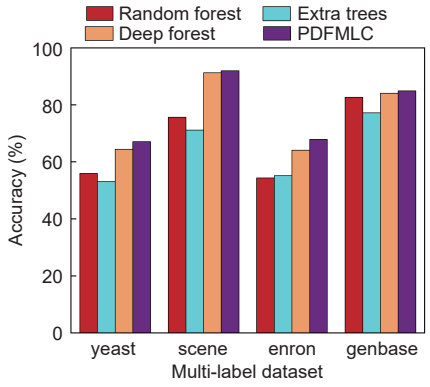


Fig. 6 Accuracy of the compared algorithms.

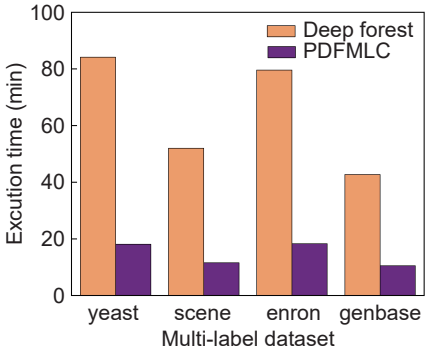


Fig. 7 Execution time of deep forest and PDFMLC.

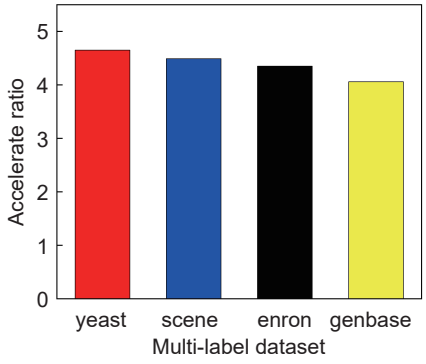


Fig. 8 Accelerate ratio of PDFMLC on four datasets.

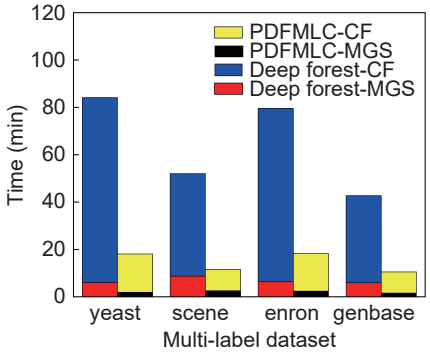


Fig. 9 Execution time of PDFMLC algorithm at MGS and CF stage.

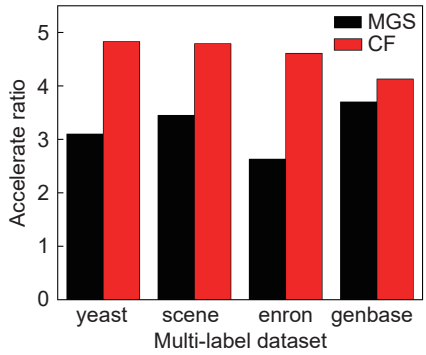


Fig. 10 Accelerate ratio of the PDFMLC algorithm at various stages.

accelerate ratio and execution time between the PDFMLC algorithm and the deep forest algorithm for the MGS forest and the CF on the benchmark datasets. Across all four standard datasets, the PDFMLC algorithm consistently achieves a speedup ratio surpassing 3.97 for the cascaded forest phase, with the highest ratio reaching 4.83 on the yeast dataset.

Notably, from Figs. 9 and 10, it is clear that the cascaded forest phase of the PDFMLC algorithm outperforms the multigranular scan forest phase in terms of speedup efficiency. This distinction can be attributed to the additional initialization operations required by the multigranular scan forest, including the transmission of training data to each node and the merging of output results from different nodes. Such operations inevitably incur network transmission latency, consequently diminishing the speedup ratio of the multigranular scan phase. In contrast, the PDFMLC algorithm leveraged the usage of broadcast variables to store the classification vector obtained in the MGS phase for the CF phase. Once transmitted initially, these broadcast variables can be subsequently re-used, effectively minimizing network transmission time and enhancing the overall speedup ratio. Generally, the scalability of an algorithm's parallelization is enhanced by increasing the scale of Spark nodes. Figure 11 illustrates the variation of algorithm runtime with the number of Spark nodes. The experiment results compare the performance of the Hadoop-based parallel deep forest algorithm, Improved Parallel Deep Forest algorithm combining with Information Theory (IPDFIT)[21] with the Spark-based parallel deep forest algorithm PDFMLC. As shown in Fig. 11, it can be observed that on the yeast dataset, when only one node is used, the values of runtime of the both compared algorithms are similar. This is because a single node cannot fully exploit the advantages of Spark's resilient delivered datasets, but still incurs resource overhead in maintaining RDD objects, spurring an extended runtime. However, as the number of nodes increases, the runtime difference between the PDFMLC and IPDFIT algorithms gradually widens. On the scene dataset, when the number of cluster nodes ranges from 1 to 5, the PDFMLC algorithm outperforms the IPDFIT algorithm, primarily due to the disk I/O operations required during the execution stages of Hadoop, as well as the need to write data to disk after each task and read data from disk for the sequence task, thereby increasing the runtime. In contrast, Spark cluster utilizes delivered resilient data storage, maintaining delivered data in memory and performing direct memory-to-memory data operations and network transfers, resulting in shorter runtime. On the enron dataset, as the number of nodes increases, the runtime difference between the two algorithms becomes more significant. On the genbase dataset, when the Spark cluster consists of five nodes, the IPDFIT algorithm exhibits longer runtime compared to that of the PDFMLC algorithm. It shows that the PDFMLC algorithm maintains a good speedup ratio as the number of nodes increases, and when the number of nodes reaches 5, a significant reduction in runtime is achieved.

Label mutual information is used to control which information is regarded in the label mutual information feature vector M and serves as the input for the cascaded forest phase. When the threshold is set high, highly correlated entity label pairs are selected. While a lower threshold enables the selection of a larger number of label pairs, it facilitates the capture of more diverse information regarding label associations.

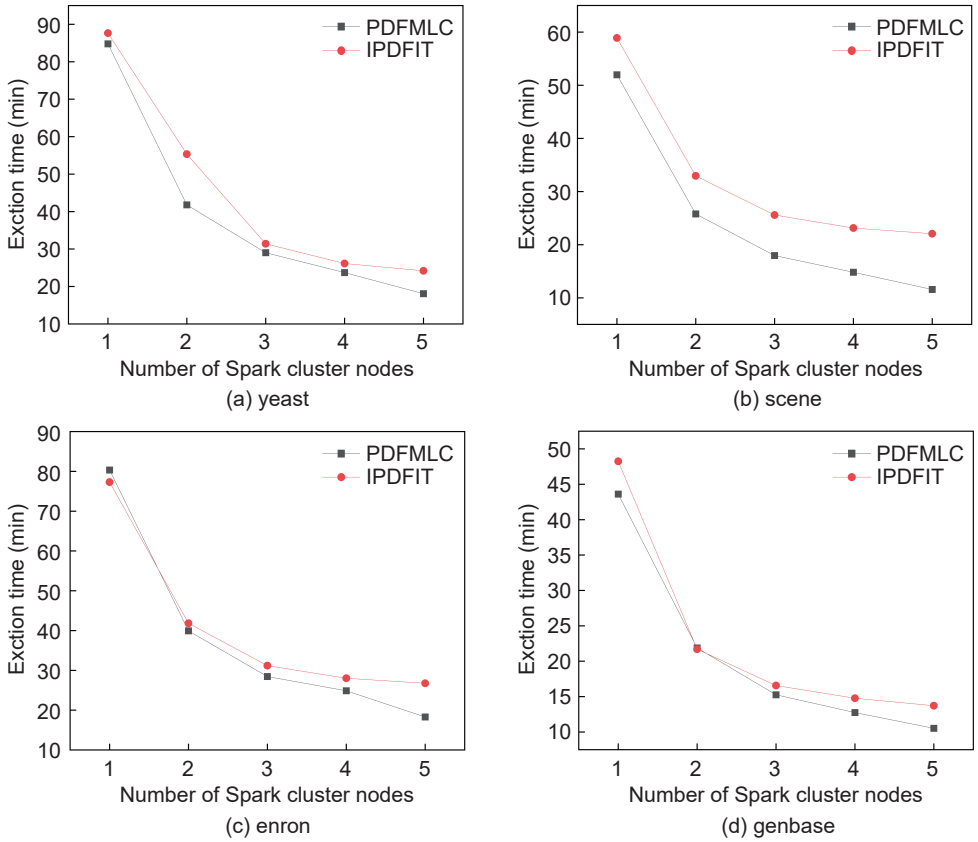


Fig. 11 Node expansion of the PDFMLC and IPDFIT algorithms.

Figure 12 shows the experiment results on the impact of label mutual information threshold on multi-label classification results. Label pairs with mutual information greater than the threshold are included in the feature vector for labeling mutual information. From Fig. 12a, it is clear that for the yeast dataset, the algorithm achieves better classification performance when the label mutual information threshold is set as 0.4. Increasing the labeling mutual information threshold from 0.4 to 0.7 does not impact the classification performance of the PDFMLC algorithm. This is because that no label pairs has mutual information values delivered within the range of 0.4 to 0.7, thus the label mutual information features selected with these thresholds are identical and do not impact the accuracy. When the label mutual information threshold is set below 0.4, label pairs with low correlation are included in the label mutual information feature vector, causing interference and resulting in poorer classification performance.

When the label mutual information threshold is set above 0.7, although highly correlated label pairs are selected, the limited number of label pairs surpassing this threshold leads to a smaller set of generated features, which does not significantly enhance the algorithm's performance, resulting in average classification accuracy. From Fig. 12b, it shows that on the scene dataset, when the label mutual information threshold exceeds 0.1, the accuracy of the tactics classification decreases. However, when the threshold is set as 0.04, the incorporated label mutual information feature improves the classification accuracy to 93.76%. From Fig. 12c, on the enron dataset, the highest accuracy for tactics classification is obtained when the label mutual information threshold is set as 0.1, incorporating a significant amount of label association information, the accuracy reaches 67.09%. When the label mutual information threshold equals or exceeds 0.2, the classification accuracy decreases as critical label pair association information is filtered out. From Fig. 12d, on the genbase dataset, a better performance is observed when the label mutual information threshold is set as 0.04, with an accuracy of 84.87%. However, when the threshold exceeds 0.1, the algorithm's classification performance slightly declines to an accuracy of 84.27%, as a large

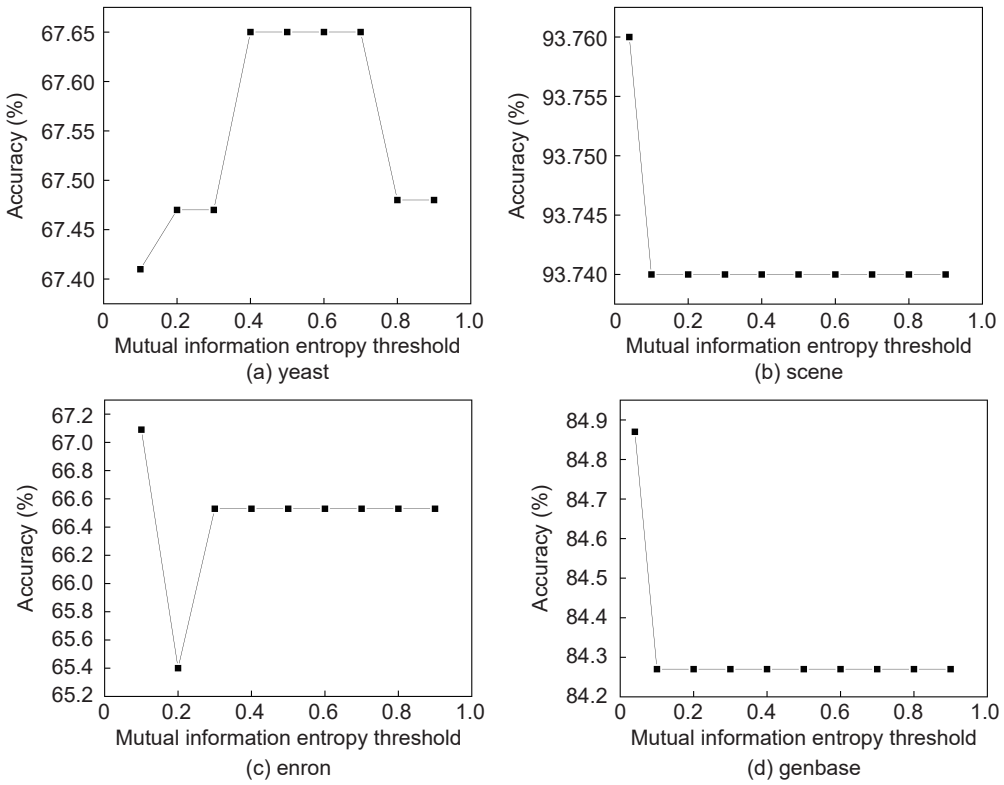


Fig. 12 Comparison of the results of different mutual information entropy thresholds.

proportion of label mutual information values fall within the range of 0 to 0.1, resulting in the filtering out of crucial association information. These experiment results indicate that in data mining of unstructured data, especially for multi-label dataset, incorporating label mutual information can enhance the learning and generalization capability of random forests.

5.5 Efficiency of PDFMLC-TIM

To validate the practicality and efficiency of the developed PDFMLC-TIM method, the comparison experiments are conducted on the intel-tactics dataset. The following algorithms are selected for comparison: random forest, rcATT[22], deep forest, extra trees, Support Vector Classification (SVC)[23], and GaussianNB. First, the cyber security analysis reports that are evaluated by the BCSE model[24] in the intel-tactics are preprocessed by converting the text into term frequency features using TF-IDF counting[13, 14]. Then, the PDFMLC algorithm is employed to classify the term frequency features and output the corresponding tactics category labels. The experiment results of the PDFMLC algorithm on the Intel-tactics are shown in Table 5.

Table 5 Performance of different algorithms on Intel-tactics.

(%)

Algorithm name	F1-score	Precision	Recall
PDFMLC	70.93	62.79	81.48
rcATT	57.38	57.77	56.99
Deep forest	70.82	64.15	79.04
Extra trees	56.19	49.98	64.17
SVC	68.39	64.32	73.02
GaussianNB	55.41	51.17	60.42

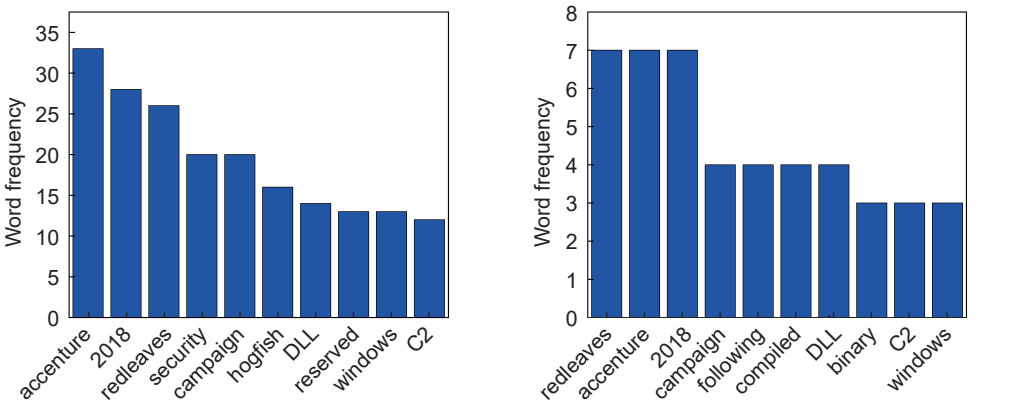
As shown in Table 5, the PDFMLC algorithm achieves the highest F1-score of 70.93% and the highest recall rate of 81.48%. This is because the PDFMLC algorithm integrates parallel multi-granularity scanning forests, parallel cascade forests, and label mutual information, which allows for better exploration of latent feature information, more detailed features, and potential correlations among features, resulting in improved tactics classification performance. In comparison, rcATT trains multiple binary classifiers for tactics category label classification, with each classifier individually classifying a specific tactics category. By treating each category independently, the algorithm overlooks the

correlations among tactics category labels. Furthermore, the Intel-tactics consists of 5 tactics category labels, requiring training of 5 binary classifiers, which increases the computation overhead. Deep forest is an ensemble learning algorithm that enhances the input instance features through multi-granularity scanning forests. It generates features of different scales using sliding windows to enrich feature information at different granularities. The stacked cascade forest structure is then utilized to learn the underlying features, where the output of the previous layer serves as the input for the next layer, ultimately predicting the classification labels. During the prediction phase, the outputs from multiple random forests are combined to derive the final predicted category. This approach effectively mitigates the negative impact of overfitting and outliers, resulting in an F1-score of 70.82%, indicating good recognition capability for tactics entity. Extra trees, which is similar to random forests, is an ensemble learning method with a high level of randomness. During training, extra trees randomly select splitting thresholds, whereas random forests search for optimal thresholds. This inherent randomness makes extra trees less prone to overfitting, and the random selection of thresholds speeds up training. However, this also leads to worse performance, as reflected in its F1-score of only 56.19%. SVC is a popular machine learning algorithm used for multi-label classification. It achieves classification by finding the optimal hyperplane that separates different class data.

In the case of the Intel-tactics, multiple binary SVC classifiers are trained, with each classifier determining whether the security text belongs to a specific tactics category. SVC performs well in tactics multi-label classification, achieving an F1-score of 68.39%, which is only 1.36% lower than that of the PDFMLC method. GaussianNB is a variant of the naive Bayes algorithm that calculates the probability of given feature values belonging to each class using Bayesian probability. GaussianNB performs poorly in Intel-tactics classification, with an F1-score of only 55.41%. These experiment results prove that the PDFMLC algorithm exhibits certain comparative advantages.

5.6 Practice of PDFMLC-TIM method

In the field of cybersecurity threat intelligence, different vendors typically provide their own cyber security analysis reports for the same cyber-attacking event. This adds to the complexity of generating threat intelligence for the attacks we encounter. In open-source data, we use the hogfish attack as an example to validate whether PDFMLC-TIM could successfully generate relevant threat intelligence for the attack. Figure 13 illustrates the term frequency distribution of two cyber security analysis reports from two different vendors regarding the Hogfish attack[25]. Reference [26] investigates the relationships between security reports. These two reports share many common high-frequency terms, such as “redleaves”, “dll”, and “C2”, “redleaves” refers to a malicious software frequently used by the Hogfish attack group in their various cyber activities. The term “dll” indicates their attack method involves Dynamic-Link Libraries (DLLs) and may include techniques like DLL injection to execute malicious code. “C2” signifies that the organization establishes command-and-control communication after compromising a host and sends malicious instructions



(a) Term frequency distribution of Hogfish in Report 1 (b) Term frequency distribution of Hogfish in Report 2

Fig. 13 Word frequency for Hogfish attack.

to the infected host. Given this reality, it is necessary to integrate threat intelligence to form a unambiguous and comprehensive threat intelligence. Here, the following approach is adopted. First, depending on the features generated by the TFIDF method, the similarity between different reports is calculated. Then, we merge the threat intelligence with the obtained high similarity. Furthermore, the proposed PDFMLC-TIM method is utilized to yield the Stix2.1-formatting security threat intelligence regarding the well-known Att@ck model for the Hogfish attack, as depicted in Fig. 14.

As shown in Fig. 14, the following information can be obtained: (1) Tactics behavior with the identifier "TA0002" corresponds to attackers attempting to execute malicious programs on devices, and a website link associated with TA0002 is provided. (2) The tactics behavior named “command and control” refers to attackers attempting to establish network channels with infected devices and send malicious commands, indicating the intent to initiate C2 attacks. (3) The website link to ATT&CK is provided for “command and control”, facilitating network security engineers to directly access information and descriptions related to C2 attacks, such as corresponding techniques and methods. Obviously, this threat intelligence is valuable in guiding network security engineers to address on detecting abnormal network communications and unusual processes on devices, thereby effectively and efficiently preventing network attack incidents, such as Hogfish attack. This also demonstrates the effectiveness of the PDFMLC-TIM method in the open-source data environment. Even when dealing with different reports of the same attack, it provides valuable threat intelligence.

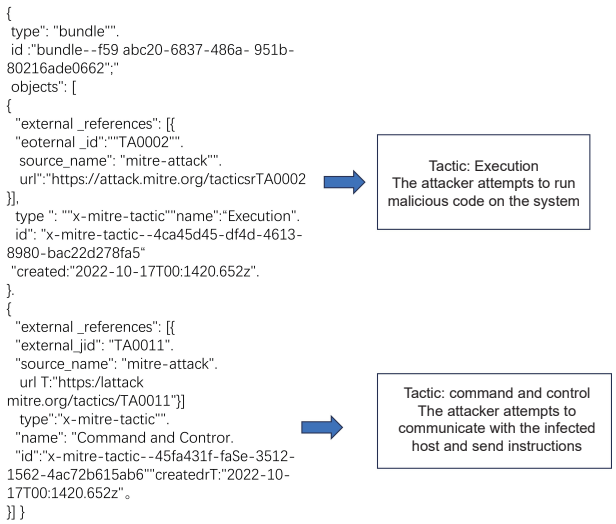


Fig. 14 Threat intelligence of Hogfish attack.

6 Conclusion and Future Work

Automated mining of tactics entities and overall threat intelligence enables seamless sharing and direct utilization of threat intelligence. This paper addresses on mining tactics entities from unstructured data in cybersecurity analysis reports and introduces the PDFMLC algorithm. The algorithm utilizes broadcast variables and the LZW algorithm to enhance acceleration. Additionally, it incorporates label mutual information into the dataset as input features, capturing intricate factors and potential label associations. Several experiments are conducted, validating the algorithm’s excellent acceleration ratio, node scalability, and superior classification capabilities. Furthermore, a PDFMLC-TIM method is developed for mining tactics entities within unstructured data. A real experiment showcase the method’s effectiveness during a public network attack event, achieving STIX2.1-formatted threat intelligence for the incident. We aim for our method to address network security issues in both the Internet of Things (IoTs) and the internet, as discussed in Refs. [27, 28]. However, it is important to note that the proposed PDFMLC algorithm and the related method currently focus solely on mining tactics, without addressing techniques and procedures. Our future research will concentrate on automating the mining process for each TTP entity and exploring the parallelism within the presented algorithm.

Acknowledgment

This work was supported by the Smart Manufacturing New Model Application Project Ministry of Industry and Information Technology (No. ZH-XZ-18 004), the Future Research Projects Funds for the Science and Technology Department of Jiangsu Province (No. BY2013015-23), the Fundamental Research Funds for the Ministry of Education (No. JUSRP211A41), the Fundamental Research Funds for the Central Universities (No. JUSRP42003), and the 111 Project (No. B2018).

References

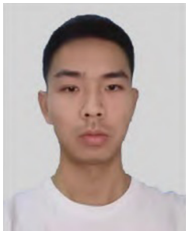
[1] N. Sun, M. Ding, J. Jiang, W. Xu, X. Mo, Y. Tai, and J. Zhang, Cyber threat intelligence mining for proactive

cybersecurity defense: A survey and new perspectives, *IEEE Commun. Surv. Tut.*, vol. 25, no. 3, pp. 1748–1774, 2023.
[2] S. M. Arıkan and S. Acar, A data mining based system for automating creation of cyber threat intelligence, in *Proc. 9th Int. Symp. Digital Forensics and Security* (*ISDFS*), Elazig, Türkiye, 2021, pp. 1–7.
[3] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, TTPDrill: Automatic and accurate extraction of threat actions from unstructured text of CTI sources, in *Proc. 33rd Annu. Computer Security Applications Conf.* (*ACSAC*), Orlando, FL, USA, 2017, pp. 103–115.
[4] W. Ge and J. Wang, SeqMask: Behavior extraction over cyber threat intelligence via multi-instance learning, *Comput. J.*, doi: 10.1093/comjnl/bxac172.
[5] MITRE ATT&CK, https://attack.mitre.org/, 2019.
[6] G. Wang, H. Peng, Y. W. Tang, and Y. Q. Jin, Error repair technology of Lempel-Ziv-Welch (LZW) compression data, (in Chinese), *Trans. Beijing Inst. Technol.*, vol. 40, no. 5, pp. 562–569, 2020.
[7] S. X. Lin, Z. J. Li, T. Y. Chen, and D. J. Wu, Attack tactic labeling for cyber threat hunting, in *Proc. 24th Int. Conf. Advanced Communication Technology* (*ICACT*), Pyeongchang, Republic of Korea, 2022, pp. 34–39.
[8] R. Rahim, M. Dahria, M. Syahril, and B. Anwar, Combination of the Blowfish and Lempel-Ziv-Welch algorithms for text compression, *World Trans. Eng. Technol. Educ.*, vol. 15, no. 3, pp. 292–297, 2017.
[9] P. E. Latham and Y. Roudi, Mutual information, *Scholarpedia*, vol. 4, no. 1, p. 1658, 2009.
[10] M. Zbili and S. Rama, A quick and easy way to estimate entropy and mutual information for neuroscience, *Front. Neuroinform.*, vol. 15, p. 596443, 2021.
[11] Curmmy, Beautiful soup documentation, https://www.crummy.com/software/BeautifulSoup/bs4/doc/index.html, 2023.
[12] PDFminer, https://euske.github.io/pdfminer/, 2014
[13] J. Deng, G. Y. Shi, T. H. Cai, J. Zhu, and L. B. Huai, Research on the method of filling of the incomplete poems of famous monks in the tang dynasty based on TF-IDF, (in Chinese), *Mod. Comput.*, vol. 25, no. 8, pp. 7–11&15, 2019.
[14] S. Kalra, L. Li, and H. R. Tizhoosh, Automatic classification of pathology reports using TF-IDF features, arXiv preprint arXiv: 1903.07406, 2019.
[15] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, MULAN: A java library for multi-label learning, *J. Mach. Learn. Res.*, vol. 12, pp. 2411–2414, 2011.
[16] E. P. Xing, Q. Ho, P. Xie, and D. Wei, Strategies and principles of distributed machine learning on big data, *Engineering*, vol. 2, no. 2, pp. 179–195, 2016.
[17] J. X. Shao, Y. N. Xing, F. Z. Nan, X. Zhao, T. H. Ma, and Y. R. Qian, Improved CK-means+algorithm and parallel implementation, (in Chinese), *Comput. Eng. Des.*, vol. 43, no. 5, pp. 1240–1248, 2022.
[18] Z. H. Zhou and J. Feng, Deep forest, arXiv preprint arXiv: 1702.08835, 2017.
[19] L. Breiman, Random forests, *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[20] P. Geurts, D. Ernst, and L. Wehenkel, Extremely randomized trees, *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.
[21] Y. Mao, J. Geng, and L. Chen, Improved parallel deep forest algorithm combining with information theory, (in Chinese), *Comput. Eng. Appl.*, vol. 58, no. 7, pp. 106–115, 2022.
[22] V. Legoy, M. Caselli, C. Seifert, and A. Peter, Automated retrieval of ATT&CK tactics and techniques for cyber threat reports, arXiv preprint arXiv: 2004.14322, 2020.
[23] S. R. Gunn, Support vector machines for classification and regression, Technical report, https://see.xidian.edu.cn/faculty/chzheng/bishe/indexfiles/new_folder/svm.pdf, 2023.
[24] F. Li, X. Yu, R. Ge, Y. Wang, Y. Cui, and H. Zhou, BCSE: Blockchain-based trusted service evaluation model over big data, *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 1–14, 2022.
[25] Proteus-Cyber, Cyber security report about Hogfish, https://proteuscyber.com/privacy-database/news/6493-abuse-of-legitimate-security-tools-and-health-sectorcybersecurity, 2022.
[26] H. Wang, K. Qin, G. Duan, and G. Luo, Denoising graph inference network for document-level relation extraction, *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 248–262, 2023.
[27] Y. Huo, J. Fan, Y. Wen, and R. Li, A cross-layer cooperative jamming scheme for social internet of things, *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 523–535, 2021.
[28] M. Moutaib, T. Ahajjam, M. Fattah, Y. Farhaoui, B. Aghoutane, and M. El Bekkali, Application of internet of things in the health sector: Toward minimizing energy consumption, *Big Data Mining and Analytics*, vol. 5, no. 4, pp. 302–308, 2022.

Zhihua Li received the BEng degree from Wuxi Light Industry University, China in 1992, and the MEng and PhD degrees from Jiangnan University, China in 2002 and 2009, respectively. He currently is a professor at Jiangnan University, China. His research interests include network technology, parallel/distributed computing, information security, edge computing, and mobile computing.

Xinye Yu received the BEng degree from Jiangnan University, China in 2022. He currently is a master student at Jiangnan University, China. His main research interest is information security.

Junhao Qian received the MEng degree from Jiangnan University, China in 1998. He currently is a full professor at School of IoT Engineering, Jiangnan University, China. His research interests include edge computing, cloud computing, and agricultural IoT engineering technology.



Tao Wei received the BEng degree from Hubei University of Technology, China in 2020. He currently is a master students at Jiangnan University, China. His research interests include parallel computing and IoT engineering technology.