Fast k-NN based Malware Analysis in a Massive Malware Environment

Jun-ho Hwang[1], Jin Kwak[2] and Tae-jin Lee[1]*
[1] Department of Information Security, College of Engineering, Hoseo University
Republic of Korea
[e-mail: hwangso93@gmail.com]
[2] Department of Cyber Security, College of Information Technology, Ajou University
Republic of Korea
[e-mail: jkwak.security@gmail.com]
*Corresponding author: Tae-jin Lee

Received November 22, 2018; revised June 11, 2019; accepted October 12, 2019; published December 31, 2019

Abstract

It is a challenge for the current security industry to respond to a large number of malicious codes distributed indiscriminately as well as intelligent APT attacks. As a result, studies using machine learning algorithms are being conducted as proactive prevention rather than post processing. The k-NN algorithm is widely used because it is intuitive and suitable for handling malicious code as unstructured data. In addition, in the malicious code analysis domain, the k-NN algorithm is easy to classify malicious codes based on previously analyzed malicious codes. For example, it is possible to classify malicious code families or analyze malicious code variants through similarity analysis with existing malicious codes. However, the main disadvantage of the k-NN algorithm is that the search time increases as the learning data increases. We propose a fast k-NN algorithm which improves the computation speed problem while taking the value of the k-NN algorithm. In the test environment, the k-NN algorithm was able to perform with only the comparison of the average of similarity of 19.71 times for 6.25 million malicious codes. Considering the way the algorithm works, Fast k-NN algorithm can also be used to search all data that can be vectorized as well as malware and SSDEEP. In the future, it is expected that if the k-NN approach is needed, and the central node can be effectively selected for clustering of large amount of data in various environments, it will be possible to design a sophisticated machine learning based system.

Keywords: k-Nearest Neighbor, Clustering, Malware

http://doi.org/10.3837/tiis.2019.12.019 **ISSN : 1976-7277**

1. Introduction

Modern cyber security threats continue to cause cross-national losses beyond individuals and corporations. Most of threats are directly or indirectly related to malicious code. In a commercial environment, malicious code is infected indiscriminately to an unspecified number of victims, or is continuously attacked by targets designated by APT attacks. In the case of malicious codes that are distributed in large quantities, the analysis difficulty is comparatively easy, but it is difficult to cope with all malicious codes due to analysis experts and financial problems. In case of APT attack, it is difficult to cope with the spread of malicious code based on unknown zero-day vulnerability of the system. In addition, malicious codes are becoming increasingly intelligent through obfuscation and logic bombs. Above all, the security industry is in an environment where a large number of malicious codes are spread, with more than one million new malicious codes on average daily. The key is that we must respond to these.

As a result, a variety of response systems have been constructed and studied to cope with malicious codes, which are key elements of security threats. Various methodologies such as signature based method and static / dynamic analysis have been proposed. Recently, machine learning based detection methodologies have been studied. Machine learning-based detection methodologies focus on prevention, not traditional post-processing. It also analyzes and classifies large amounts of malicious code through automated processing of appropriate machine learning algorithms. On the other hand, malicious codes can be regarded as unstructured data in general, which makes it difficult to handle outliers in case of machine learning based malicious code classification.

The k-NN algorithm has the advantage of being less influenced by the tendency or outliers of the data because it is classified into k adjacent label values for Training data. In addition, simple algorithms result in intuitive results. k-NN is also advantageous in terms of explainable AI because complex algorithms such as DNN/RNN can not explain the process of deriving the result. However, the major disadvantage of the k-NN algorithm is that there is a problem of computation speed as the number of data to be compared increases as the training data increases. In this paper, we propose a Fast k-NN algorithm to mitigate the computation speed problem of the k-NN algorithm.

In Section 2 describes related research on SSDEEP and k-NN-based malicious code analysis, which compares the similarity of data. Section 3 describes the core algorithm of Fast k-NN proposed in this paper. In Section 4, we compare the performance of the k-NN and the proposed Fast k-NN algorithm. Section 5 describes the meaning and conclusion of the Fast k-NN algorithm proposed in this paper.

2. Related Work

2.1 Similarity based Malware Analysis

In a large number of malicious code environments, malicious code is distributed mainly to indefinitely infect large numbers of malicious codes to an unspecified number of victims. In a commercial environment, malicious code is mainly analyzed based on signatures due to APT attacks, but it is a great burden for analysts to analyze all the malicious codes that are being distributed. Therefore, malicious codes are analyzed and classified by signature based

classification, hash value based classification, and similarity based classification using the database constructed by analyzing existing malicious codes. In case of similarity - based classification, malicious code can be classified through similarity between data even though it does not completely coincide with existing data. In addition, similarity-based classification is relatively free of variables such as data tendency and data amount compared with pattern matching method by setting an appropriate threshold value for similarity value. Because of this advantage, similarity-based malicious code analysis research has been carried out variously. SSDEEP is a typical malicious code measurement method.

SSDEEP is a similarity measure algorithm based on fuzzy hashing. Fuzzy hashing is an algorithm proposed by Jesse Korblum in 2006. Unlike general hashing algorithms such as MD5 and SHA256, it consists of piecewise hashing for dividing data into N-block units and Rolling hash for calculating hash values of high-speed block units. partial matches can be used to identify the similarity of data in adjacent block units. Various kinds of researches are being conducted to classify malicious codes through SSDEEP instead of the signature of malicious code in a large number of malicious code environments, or in cases where binary data is mostly similar, such as variant malicious codes.

Y. Li proved that the Fuzzy hashing algorithm is practically applicable to malicious code similarity analysis, and that it can be practically used for malicious code similarity analysis by evaluating the fuzzy hashing algorithms in the public malware dataset as a comprehensive framework [1]. AP Namanya proposed a proof-of-concept technique to detect similarity of files based on a similarity percentage between known and unknown to reduce the effect of obfuscation techniques of fuzzy hashing [2]. S. Gupta proposed a framework for linking malicious code with the Fuzzy Hashing algorithm by constructing high-level Category Sequences by extracting Windows API Call Sequences for each malicious code group and mapping the API to 26 categories [3]. **Fig. 1** Framework diagram proposed by S. Gupta.

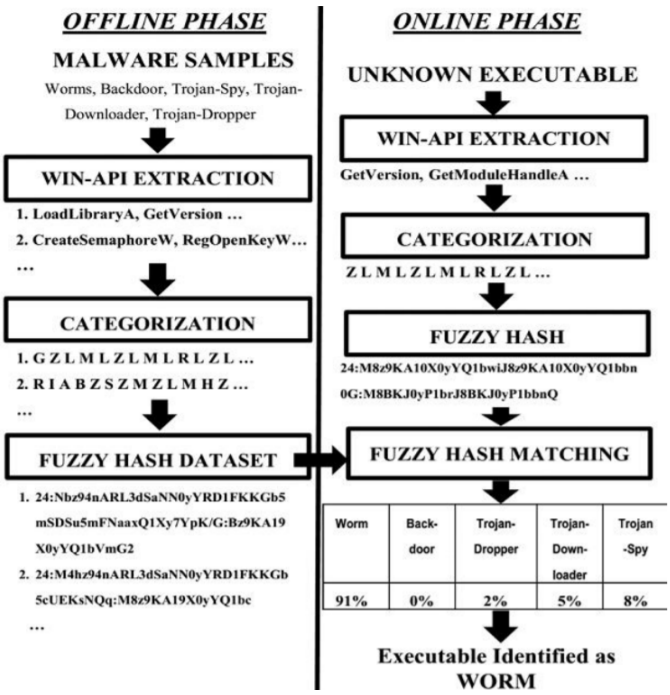

Fig. 1. S. Gupta's Malware Classification Framework

In the study of classifying malicious codes by using SSDEEP, it is most common to identify the label through fuzzy hash matching such as **Fig. 1** Fuzzy Hash Matching is compared with Hash of Unknown file to be classified using Fuzzy Hash Dataset or training data constructed in advance like **Fig. 1** OFFLINE PHASE. At this time, in order to determine the label of the Unknown file, it is the same as selecting the k value of the k-NN algorithm to decide how many similar files to use in the training data. In conclusion, many existing studies [7] [8] [9] that determine label by comparing with learning data when classifying malicious code using SSEEP are based on k-NN algorithm. As a result, although the detection performance generally improves as the learning data increases, there is a problem that the time complexity required for the final classification increases inevitably.

2.2 k-NN based Malware Analysis

In order to cope with a large number of malicious code environments, researches on classification of malicious codes using machine learning algorithms are actively conducted. However, since malicious codes can be regarded as unstructured data in general, there is a difficulty in handling outliers in the case of machine learning based malicious code classification. The k-NN algorithm is an intuitive and simple method for classifying neighboring k neighbors.

The k-NN algorithm has the advantage of being less influenced by the tendency or outliers of the data because it is classified into k adjacent label values for training data. k-NN is also advantageous in terms of explainable AI because complex algorithms such as DNN/RNN can not explain the process of deriving the result.

On the other hand, as training data increases, the time complexity increases by *O(n)*. Because of this feature, many studies of k-NN algorithm related to high-speed processing have been carried out considering the total performance comparison problem [10] [11] [12]. J Chen proposed Two Divide and Conquer Methods for computing approximate k-NN for high-order data [4]. C. Yu proposed an effective search process by indexing distances to find approximate data in the k-NN algorithm [5]. Z. Yong proposed a k-NN algorithm by setting representative points through clustering of data [6]. Z. Young tried to solve the problem of comparing the total number of k-NNs in an unstructured data environment through clustering. However, he pointed out that the distance between samples in the same category is larger than the sample distance in the other category due to the uneven distribution of data among the categories and not compactness. For example, as in **Fig. 2**, the distance between d_2 and d_3 of C_k is longer than the distance between d_1 and d_2 of C_j. Therefore, he proposed an algorithm that selects a center node with the k-mean algorithm like **Fig. 2** and then remove points that are moderately away from the center node. If the gray circle of appropriate size for each cluster is drawn in **Fig. 3**, the outer data of the circle, d_1, d_2, d_3, d_4 and d_5 are removed and the operation speed of k-NN can be improved. In other words, when samples for training data are clustered, it effectively reduces some data showing an inaccurate peak, and reduces the burden on k-NN's total similarity calculation time. However, this process is ambiguous in that the model design is unpredictable and the criteria for uncertain data. It can be operated inappropriately if it is applied to the malicious code classification domain.

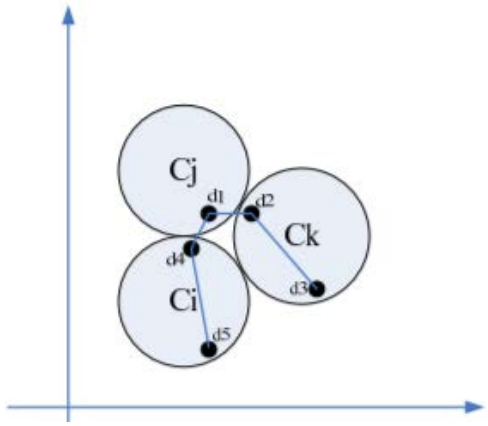

Fig. 2. Z. Yong's Multi-peak Distribution of Samples

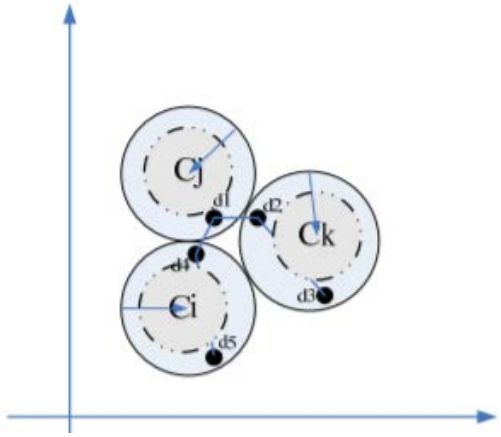

Fig. 3. Z. Young's Distribution After Sample Austerity

3. Proposed Model

3.1 Overview

The main process of the Fast k-NN algorithm proposed in this paper can be divided into two steps. The clustering step is a process that makes the proposed methodology applicable to the learning data applicable. The fast nearest neighbor search method is a process of applying a practical methodology to cope with a large increase in training time, which is a main disadvantage of the k-NN algorithm.

3.1.1 Clustering Step

The main process in the clustering phase is to cluster the training data. The advantage of clustering of training data is that the time complexity needed for clustering is added in the training phase, but the relationship between training data can be identified preferentially. Also, even in the case of clustering time complexity, it may be a burden for a server providing a model from a commercial environment point of view, but it is not a sensitive issue for a client that performs only a test with a generated model.

The proposed clustering scheme in this paper does not depend on the algorithm using only a clustering algorithm with a central node. For example, in this paper, we use a modified method of selecting arbitrary node as a central node in the single linkage method based on the SSDEEP similarity in order to derive the test data result, but there are several clustering methods (k-Means, k-Medoids, etc.).

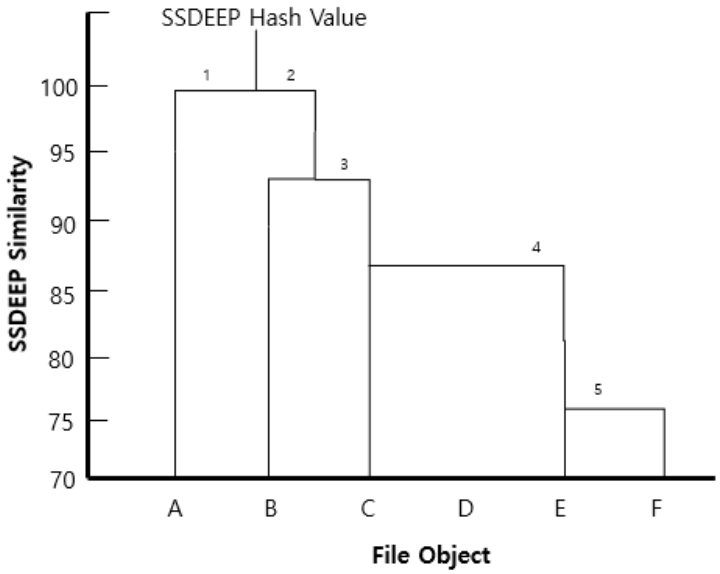

Fig. 4. SSDEEP Similarity Based Single Linkage Clustering

The clustering method used in this paper is processed in the following two phases.

(1) Training data SSDEEP Similarity-based Single Linkage clustering
- One or multiple cluster groups are derived through SSDEEP similarity based single linkage clustering of Training data.

(2) Random central node selection and group node alignment
- Central nodes are selected randomly in the clustering group, and grouped by group node and group node not similar to central node by using SSDEEP similarity and managed by file

As a result, data can be managed by clustering the learning data and arranging the cluster nodes in order of the cluster nodes that are not similar to the central node for each cluster group based on the central node. In this process, the process of managing data by sorting in the order of dissimilar group nodes operates as a core process in order to operate the Fast k-NN algorithm proposed in this paper.

3.1.2 Fast Nearest Neighbor Search Step

If clustering is used to group the learning data on the basis of similarity, The distance $d(C_0,P_0)$ between the central node C_0, C_1, $C_2,\ldots,C_{n-1},C_n$(n is the number of groups) and the group nodes $P_0,P_1,P_2,\ldots,P_{m-1},P_m$($m$ is the number of nodes in the group) excluding the central node can be calculated by preceding the clustering group. In this case, we can use several similarity measurement techniques for distance d. In this paper, SSDEEP similarity is used. Therefore, the distance d can be obtained by measuring the similarity of SSDEEP between C_0 and P_0 files and follow the formula below.

$$d(C_n,P_m) = SSDEEP\ Similarity(C_n,P_m), n>0, m>0 \quad (1)$$

The clustered learning data can be represented as **Fig. 5** when the analysis target file *Q* is located at a certain point.

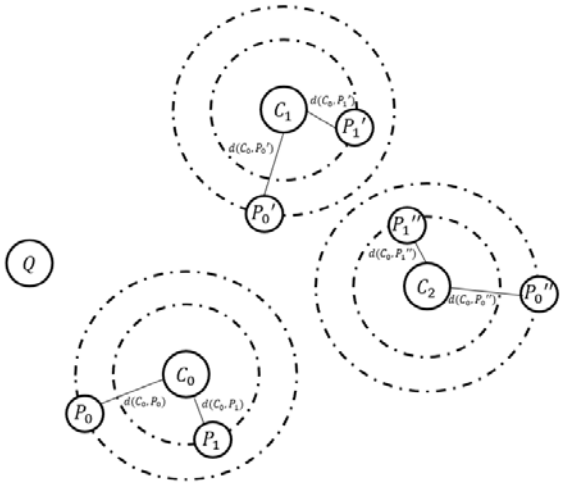

Fig. 5. Examples of central node and group node relationships

Fast nearest neighbor search method is a key process of Fast k-NN, and it is a methodology to inspect objects *Q* and group nodes $(P_0, P_1, P_2, \ldots, P_{m-1}, P_m)$ that do not need similarity calculation based on similarity between learning data measured in advance. This can be done in the following four processes.

a. The closest clusters are checked in order
First, the cluster closest to the analysis target file *Q* among the clustered learning data starts to be examined. In this case, the closest clusters are in the descending order of similarity with the central node $C_0, C_1, C_2, \ldots, C_{n-1}, C_n$(*n* is the number of groups) of each cluster. By checking the closest clusters, the process can work effectively for the purpose of searching the most similar *k* files. When the learning data is divided into three clusters such as **Fig. 6** and the similarities of the central nodes of each cluster are $d(C_0, Q)$=70, $d(C_1, Q)$=50, $d(C_2, Q)$=40, the cluster order of C_0, C_1 and C_2 nodes is examined.

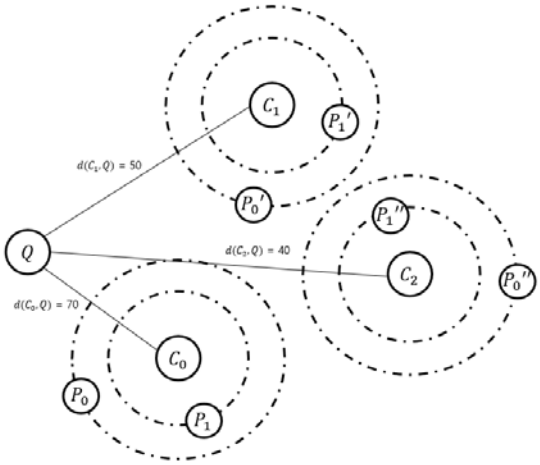

Fig. 6. Examples of analysis node and center node relationships

b. From the central node, check the least similar group nodes

Second, if a cluster group to be checked is specified, check the group node $(P_0, P_1, P_2, \ldots, P_{m-1}, P_m)$ that is the closest to the central node among the clusters. Assuming that there is a group node (P_0, P_1, P_2) in a cluster group such as **Fig. 7**, when the similarity is measured with the central node, the distance is checked in the order of P_0, P_1, P_2 which are the most distant (not similar).

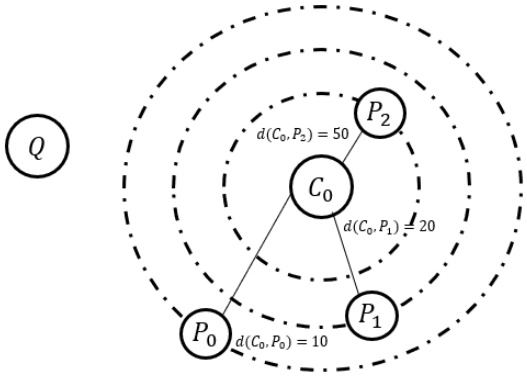

Fig. 7. Example of check sequence in a cluster

c. Exclude nodes beyond a certain distance

Third, basically, similarity calculation for total comparison of all data is the main cause of performance degradation of k-NN algorithm. Therefore, it is possible to improve the performance of k-NN efficiently by excluding the nodes that are not within a certain distance from the check by a simple method when starting the inspection from the group node which is not similar to the previous process. As shown in **Fig. 8**, assuming that the file Q, the central node C_0, and the group node (P_0, P_1, P_2) exist, the minimum distance Q and P_0 can have is $d(C_0, Q) - d(C_0, P_0)$. Similarly, the minimum distance Q and P_1 can have is $d(C_0, Q) - d(C_0, P_1)$. This is shown in **Fig. 8**.

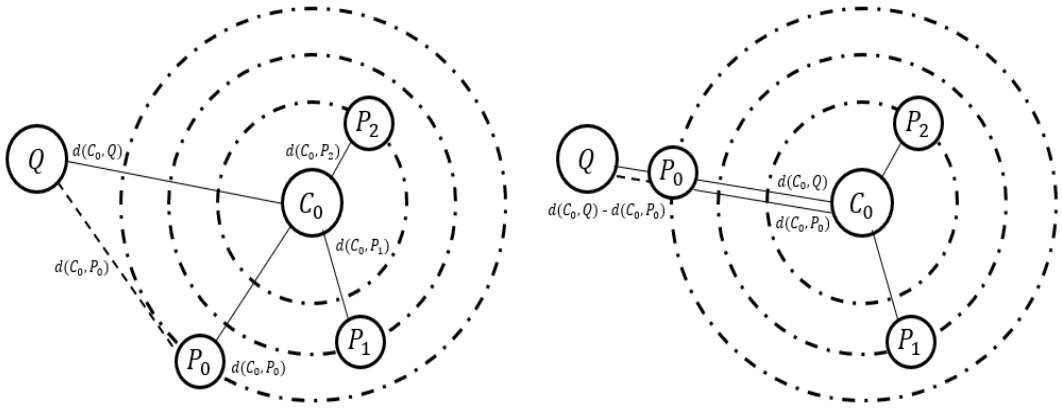

Fig. 8. Minimum distance $d(Q, P_0)$

Fast k-NN is effective method to improve algorithm performance. For example, as shown in **Table 1**, it is assumed that the similarity between the analysis target file Q and each node is a real distance vector, and the maximum similarity that the analysis target file Q can have is Max distance vector. If the data is classified into k = 1, the similarity value of 35, which is the maximum similarity value that P_2 can have, is not higher than 70, which is the similarity value of P_0. Therefore, it can be determined that the similarity calculation of P_2 and Q need not be performed.

Table 1. Similarity calculation decision table

Node	Real distance vector	Max distance vector
P_0	70	90
P_1	50	80
P_2	30	35

The proposed Fast k-NN algorithm has previously calculated $d(C_0,P_0), d(C_0,P_1), d(C_0,P_2) \ldots d(C_0,P_m)$ by preceding clustering of training data. Therefore, the similarity calculation is performed between the analysis target file Q and the central nodes of the respective clusters, and it is determined whether the operation is to be performed on each group node. Also, in the above example data, the k-NN algorithm needs to perform 3 similarity calculations, but the Fast k-NN algorithm can produce the same result with only 2 similarity calculations. Generally, a large amount of data will be composed in one cluster, so it will work more effectively in a real environment.

d. Check the rest of the cluster group

Fourthly, if one cluster is checked whether all nodes need to be computed or computed, it starts checking the cluster with a similar central node after the cluster. At this time, if the distance between the central node and the central node and the closest non-similar node in the same cluster is lower than the similarity of the *k*-th node as in the above-described method, it can be judged that the corresponding cluster need not be entirely examined.

The entire operation sequence of Fast k-NN can be expressed as follows **Fig. 9**.

```
READ q
INIT max distance, k-pair
FOR cluster group IN cluster group size
  FOR node IN cluster size
    IF d(q, center node) − d(node, center node) > max distance THEN
      CONTINUE
    ELSE
      IF d(q, node) < max distance
      THEN
        UPDATE max distance
        UPDATE k-pair
    End
  End
End
```

Fig. 9. Fast k-NN process

3.2 System Configuration Diagram

Based on the above-mentioned proposals, the designed system is shown in **Fig. 10**.

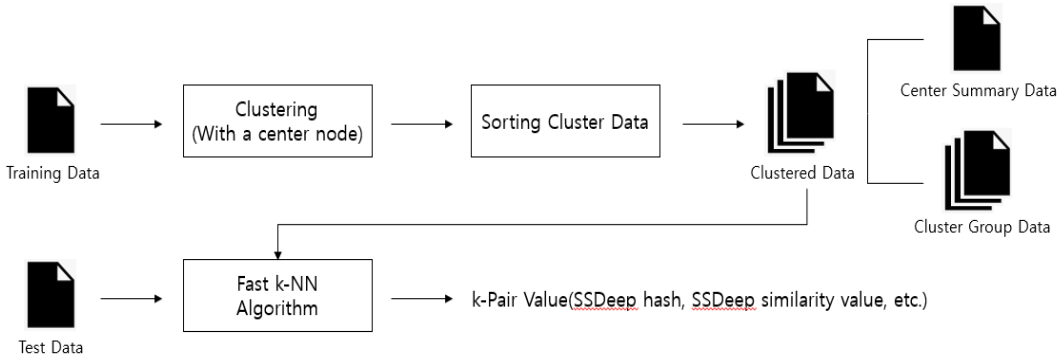

Fig. 10. Proposed System Configuration Diagram

In the training phase, the training data is clustered, the center nodes selected for each cluster are managed by Center Summary Data, and the group nodes in the same cluster are sorted and arranged in descending order of similarity with the central node. The test data can be analyzed at high speed using the cluster data and the Fast k-NN algorithm.

4. Experiment

The Fast k-NN proposed in this paper operates basically in linux environment and tested in virtual environment of windows host. We configured a virtual environment of 2GB RAM and ubuntu 16.04 environment on 1.80GHz dual core CPU, 8GB ram and windows 10 environment host. Section 4.1 describes the dataset used in this paper and Section 4.2 describes the performance measurement using Fast k-NN.

4.1 Dataset

A total of 6,526,885 data were clustered into 1,237 clusters. It uses a single-linkage clustered data set provided by Virusshare [13], an open malware collection channel. The training data is divided into several data sets for testing and is shown in **Table 2** below. As a test data, 100 data of some clusters in the data were randomly selected and used.

Table 2. Dataset Configuration

Dataset category	Number of SSDEEP	Number of clusters
Dataset-1	6,526,885	1,237
Dataset-2	2,997,380	300
Dataset-3	202,443	40
Dataset-4	103,251	25

4.2 Fast k-NN Performance Measurement

The test values were performed from the point of view of the number of similarity calculation comparisons to classify 100 test data of k-NN and Fast k-NN. **Table 3** and **Table 4** shows the

number of comparisons of average similarity required for classification of k-NN algorithm for each data set.

Table 3. The number of k-NN and Fast k-NN similarity comparison operations(k=5)

Dataset category	k-NN	Fast k-NN	Fast k-NN/k-NN
Dataset-1	6,526,885	19.71	0.0003%
Dataset-2	2,997,380	14.82	0.0005%
Dataset-3	202,443	9.98	0.0049%
Dataset-4	103,251	7.94	0.0077%

Table 4. The number of k-NN and Fast k-NN similarity comparison operations(k=3)

Dataset category	k-NN	Fast k-NN	Fast k-NN/k-NN
Dataset-1	6,526,885	12.93	0.0002%
Dataset-2	2,997,380	9.83	0.0003%
Dataset-3	202,443	6.54	0.0032%
Dataset-4	103,251	5.04	0.0049%

In this way, the fast k-NN algorithm proposed in this paper can solve the problem of the time complexity that increases as the number of data increases due to the total number comparison of k-NN. As a result, the similarity calculation of SSDEEP took 0.001 seconds to compare one data. Therefore, the number comparison of k-NN requires a large number of similarity calculation. In the case of Fast k-NN, the time required for the calculation and the number of comparisons were increased as the k value increased. However, in the test environment, the number of comparisons was 2.0e-6 the performance of k-NN can be greatly improved.
Table 5 shows the comparison results of k-NN algorithm and Fast k-NN algorithm per 100,000 data sets for each data set.

Table 5. The number of k-NN and Fast k-NN similarity comparison operations per 100K dataset

Dataset category	Num of data	Number of comparisons per 100K	
		Fast k-NN(k=3)	Fast k-NN(k=5)
Dataset-1	6,526,885	0.1983	0.3023
Dataset-2	2,997,380	0.3288	0.4957
Dataset-3	202,443	3.2700	4.9900
Dataset-4	103,251	5.0400	7.9400

In the **Table 5**, it can be seen that the number of comparison operations per 100,000 pieces decreases as the number of data increases. Dataset-1 is more than 25 times more efficient than Dataset-4 in terms of number of comparisons per 100,000. In conclusion, the Fast k-NN algorithm works more efficiently as the size of the dataset increases and as the value of k decreases. It is considered that these test results can improve the disadvantage of the k-NN algorithm, which is drastically degraded as the training data increases.

5. Conclusion

The current security industry is in an environment where a large number of malicious codes are distributed indiscriminately. Malicious code research using machine learning algorithms is meaningful as a precautionary measure rather than a post - treatment to cope with an overwhelming number of malicious codes. On the other hand, malicious codes can be regarded as unstructured data in general, and there is a difficulty in handling outliers in case of

machine learning based malicious code classification. The k-NN algorithm has the advantage of less influence on the tendency or outliers of data compared to other algorithms by classifying the training data into k adjacent label values. However, the major disadvantage of the k-NN algorithm is that the time complexity increases significantly as the learning data increases. Therefore, in this paper, we propose a Fast k-NN algorithm to mitigate the computation speed problem of the k-NN algorithm. It is expected that the advantages of unstructured data and computation speed problem will be solved at a certain level.

When the proposed Fast k-NN algorithm is evaluated using 6,526,885 Virusshare datasets, it can be processed with 2.0e-6 comparisons than the computation required for total comparison based on the tested environment. The proposed Fast k-NN algorithm has advantages of the k-NN algorithm and greatly improved the computation speed. It can also be used to search all data that can be vectorized as well as malware and SSDEEP. In the future, it is expected that if the k-NN approach is needed, and the central node can be effectively selected for clustering of large amount of data in various environments, it will be possible to design a sophisticated machine learning based system.

Acknowledgement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2018R1C1B5029849) and by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(No. NRF-2017R1E1A1A01075110).

References

[1] Li, Yuping, et al, "Experimental study of fuzzy hashing in malware clustering analysis," in *Proc. of 8th workshop on cyber security experimentation and test (cset 15), USENIX Association Washington, DC*, vol. 5, no. 1, 2015. Article (CrossRef Link)

[2] Namanya, Anitta Patience, et al, "Detection of malicious portable executables using evidence combinational theory with fuzzy hashing," in *Proc. of Future Internet of Things and Cloud (FiCloud), IEEE 4th International Conference on. IEEE*, 2016. Article (CrossRef Link)

[3] Gupta, Sanchit, Harshit Sharma, and Sarvjeet Kaur, "Malware Characterization Using Windows API Call Sequences," in *Proc. of International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Cham*, pp. 271-280, 2016. Article (CrossRef Link)

[4] Chen, Jie, Haw-ren Fang, and Yousef Saad, "Fast approximate kNN graph construction for high dimensional data via recursive Lanczos bisection," *Journal of Machine Learning Research*, vol. 10, pp. 1989-2012, Sep, 2009. Article (CrossRef Link)

[5] Yu, Cui, et al, "Indexing the distance: An efficient method to knn processing," *Vldb*, vol. 1, 2001. Article (CrossRef Link)

[6] Yong, Zhou, Li Youwen, and Xia Shixiong, "An improved KNN text classification algorithm based on clustering," *Journal of computers*, pp. 230-237, 2009. Article (CrossRef Link)

[7] Dunham, Ken, "A fuzzy future in malware research," *The ISSA Journal*, pp. 17-18, 2013. Article (CrossRef Link)

[8] Raff, Edward, and Charles Nicholas, "Lempel-Ziv Jaccard Distance, an effective alternative to ssdeep and sdhash," *Digital Investigation*, vol. 24, pp. 34-49, 2018. Article (CrossRef Link)

[9] Hiruta, S., et al, "Evaluation on malware classification by combining traffic analysis and fuzzy hashing of malware binary," in *Proc. of he International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*, 2015. Article (CrossRef Link)

[10] Liu, Y. D., and H. M. Niu, "KNN classification algorithm based on k-nearest neighbor graph for small sample," *Computer Engineering*, pp. 198-200, 2011. Article (CrossRef Link)
[11] Weinberger, Kilian Q., and Lawrence K. Saul, "Fast solvers and efficient implementations for distance metric learning," in *Proc. of the 25th international conference on Machine learning. ACM*, pp. 1160-1167, 2008. Article (CrossRef Link)
[12] Li, Shengqiao, E. James Harner, and Donald A. Adjeroh, "Random KNN feature selection-a fast and stable alternative to Random Forests," *BMC bioinformatics*, Article number. 450, 2011. Article (CrossRef Link)
[13] VirusShare.com - Because Sharing is Caring. https://virusshare.com/
[14] A. Lakhotia, A. Walenstein, C. Miles, A. Singh, "VILO: A Rapid Learning Nearest-neighbor Classifier for Malware Triage," *Journal in Computer Virology*, vol. 9. no. 3. pp. 109-123, 2013. Article (CrossRef Link)
[15] V. Harichandran, F. Breitinger, I. Baggili, "Bytewise Approximate Matching: The Good, The Bad, and The Unknown," *Journal of Digital Forensics, Security and Law*, vol. 11, no. 2, 2016. Article (CrossRef Link)
[16] F. Breitinger, G. Stivaktakis, H. Baier, "FRASH: A framework to test algorithms of similarity hashing," *Digital Investigation*, vol. 10, pp. S50-S58, 2013. Article (CrossRef Link)
[17] H.-S. Park, C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Systems with Applications*, vol. 36, pp. 3336-3341, 2009. Article (CrossRef Link)
[18] Y. Ye, T. Li, Y. Chen, Q. Jiang, "Automatic malware categorization using cluster ensemble," in *Proc. of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM*, pp. 95- 104, 2010. Article (CrossRef Link)
[19] S. Pai, F. Di Troia, C. A. Visaggio, T. H. Austin, M. Stamp, "Clustering for malware classification," *J Comput Virol Hack Tech*. vol. 13, no. 2, pp. 95-107, May 2017. Article (CrossRef Link)
[20] M. Asquith, "Extremely scalable storage and clustering of malware metadata," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no 2, pp. 49-58, May 2016. Article (CrossRef Link)
[21] J. Saxe, K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. of Malicious and Unwanted 47 Software (MALWARE), 2015 10th International Conference on, IEEE*, pp. 11-20, 2015. Article (CrossRef Link)
[22] G. E. Dahl, J. W. Stokes, L. Deng, D. Yu, "Large-scale malware classification using random projections and neural networks," in *Proc. of Acoustics, Speech and Signal Processing (ICASSP), IEEE*, pp. 3422-3426, 2013. Article (CrossRef Link)

Jun-Ho Hwang Author Hwang received bachelor's degree in information security from Hoseo university. Currently, the master's course is in progress from hoseo university in information security. His research is mainly focused on malicious code analysis, machine learning, image processing.

Jin Kwak is a professor at Dept. Of Cyber Security in Ajou University, Korea. He received the Ph.D. degree from SKKU, Korea. His research interests include Cryptographic protocols, Applied security mechanisms for Cloud and Big Data system and so on.

Tae-Jin Lee Professor Lee received bachelor's degree in computer science from POSTECH, master's degree from Yonsei University, and doctor's degree. He worked in Korea Information Security Agency for a long time and conducted cyber security research. His research is mainly focused on malicious code analysis, anomaly analysis, and system security.