*International Journal of Engineering & Technology*, 7 (3.33) (2018) 23-27

International Journal of Engineering & Technology

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***


# A Study on the Performance of Feature Extraction Methods According to the Size of N-Gram

**Young-Man Kwon[1], Min-Gu Son[2], Dong-Keun Chung[3] and Myung-Jae Lim[4*]**

[1]*Department of Medical IT, College of Healthy Industry, Eulji University, Korea*
[2]*Department of Medical IT, College of Healthy Industry, Eulji University, Korea*
[3]*Department of Medical IT, College of Healthy Industry, Eulji University, Korea*
[4]*Department of Medical IT, College of Healthy Industry, Eulji University, Korea*
**Corresponding author E-mail: lk04@eulji.ac.kr*

**Abstract**

In this paper, we studied the performance of feature extraction methods according to the size of N-gram for malware detection. The feature is extracted by three methods, using Opcode Only, both Opcode and API and API Only from PE file. We measure the performance of them indirectly with measuring the AUC score and accuracy of classifier. We did experiments with the different N size by using several classifiers such as DT, SVM, KNN and BNB classifiers. As a result, we got the conclusion as followings. If we use N-gram technique, we recommend Opcode Only method through our experiments. Also, the instance-based classifier KNN and DT among the model based classifier have good performance than SVM and BNB.

***Keywords***: *malware detection, machine learning, classifier, N-gram, Opcode, API*

## 1. Introduction

Malware, also called malicious software, is the software that damage systems by intension. In last 10 years, according to AC-TEST's research, the number of malwares is rebounded explosively [1]. Therefore, malware detection technique needs to be more powerful to deal with enormous and unknown malwares. In malware detection domain, we can classified traditional malware detection methods roughly as signature-based and behavior-based [2]. Signature-based method define whether the file is malware or not by comparing with collected signature that was detected from malwares in past. However, maintaining and updating signatures require high cost and it is hard to deal with the malware which has new signature. Behavior-based method decide whether the file is malware or not by executing it and monitor its behavior. But this method needs high cost and require specific environment to handle the malware. Both methods have each pros and cons.
However, these traditional methods are not enough to deal with next generation threats [3]. Therefore, recently many researchers utilize other techniques such as machine learning for malware detection [4]. In this paper, we made feature vector by applying N-gram feature extraction technique additionally for the three feature extraction methods those are Opcode only, both Opcode and API, API only. Then, we measured the performance of them indirectly with measuring the AUC score and accuracy of classifier.
In this paper, we use several classifiers during experiment to measure the performance. Those are Decision Tree, K-Nearest Neighbor, Support Vector Machine and Bernoulli Naïve Bayes classifier.

## 2. Related Work

### 2.1. Feature Extraction Methods

There are several feature vectors of execution file for malware detection. In the case of Window OS, the format of execution file is the type of PE file. From PE file, we can get various features such as PE header, Opcode, Window API calls and so on.
Edward Raff used MS-DOS, COFF and Optional headers that are extracted from PE header to compare domain knowledge method and domain knowledge-free method by using raw data [5]. Yibin Liao used checksum, section name, initialized data size, DLL characteristics, major image version and so on [6]. Igor Santos used Opcode sequence to detect variants of known malware families [7]. Veermani used API calls based on that different malware family have different API calls [8]. Ahn proposed feature vector that used combination of Opcode and Window API calls and showed better performance than only used Opcode or Windows API calls [9].

### 2.2. N-Gram

The N-gram technique is the sequence of given text or speech that split into combinations of size N. It also is called 'unigram', 'bigram' and 'trigram' when the size of N is 1, 2 and 3. N-gram considers before and after of the words so, it can capture the meaningful feature. The N-gram is effectively used in Natural Language Processing domain. Willeam used N-gram for text categorization tasks and got high accuracy 99.8% [10]. Payal used unigram to trigram for sentimental analysis on tweets [11]. Also, N-gram is widely used in malware detection domain. Mikhail used N-gram

model to extract essential feature from operation code sequence and construct an N-gram frequency vector [12].
The example of N-gram with Opcode is shown in fig. 1.

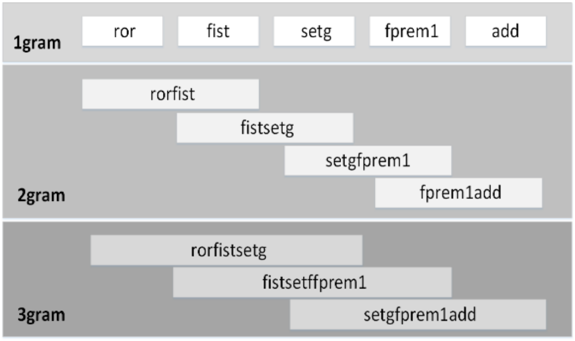



**Fig. 1:** The example of N-gram Opcode

## 2.3. Classifier

We measured the performance of classifiers according to the different size of N over the three feature extraction methods. Measuring the performance of classifiers with different N-gram means that we measure the performance of feature extraction methods according to N-gram indirectly. In our experiment, we used DT (Decision Tree), KNN (K-Nearest Neighbors), SVM (Support Vector Machine) and BNB (Bernoulli Naïve Bayes) classifier.

### 2.3.1. Decision Tree

DT is the tree based model that is widely used for classification and regression. DT can be visualizing and it is easy to interpret the decision process. It consists of root node, nodes and leaf nodes. Each node is split into 2 subsets by each decision rule until it becomes leaf node.
In classification task, each leaf nodes represent the class label. The goal of DT is learning the simple decision rules to predict the target value. The decision rule is based on impurity. During learning, DT makes decision rule to minimize impurity as possible. There are several indices to measure the degree of impurity such as Gini, entropy. The equation of Gini which is used commonly is given by [13].

$$H(X_m)= \sum_k p_{mk} (1- p_{mk}) \tag{1}$$

Where, the $p_{mk}$ is the proportion of class k observations in node m. When training DT, the CART (Classification and Regression Tree) algorithm is commonly used. It split the node through a single feature k and its threshold $t_k$ [14]. The equation of cost function that CART algorithm tries to minimize is given by.

$$J(k, t_k)= \frac{m_{left}}{m} G_{left}+\frac{m_{right}}{m} G_{right} \tag{2}$$

Where, $G_{left/right}$ is the impurity of left and right subset and $m_{left/right}$ is the number of instances in the left and right subset.

### 2.3.2. K-Nearest Neighbors

KNN is the instance-based, also called lazy learning algorithm that do not train discriminative function and just arranges the train data. In classification task, it calculates the distance between new data and train data. Then, found number of k-neighbors and classify the new data with majority vote by them. KNN require high computational cost if the dataset is large. There are several methods to calculate the distance. The most commonly used method is Euclidian distance. Also, setting the appropriate value of K is important in KNN. If the value of K is too small, it can be highly influence by the noise. In contrast, if the value of K is too big, it makes high computational cost and the low performance.

### 2.3.3. Support Vector Machine

SVM is powerful model that can handle many tasks such as pattern recognition, linear or nonlinear classification, regression and outlier detection. The goal of SVM is to make maximum margin in decision boundary which is create by few instances of train data that location in between two classes' boundary. Those data points are called support vectors. SVM classifies new instance by calculating the distance between support vectors and new instance. Given a training set of instance-label pairs $(x_i, y_i)$, i=1, … , l where $x_i \in R^n$ and $y \in \{1, -1\}^i$, SVM require the solution of the following optimization problem [15].

$$\min_{w,b,\xi} \quad \frac{1}{2}w^T w+C\sum_{i=1}^{l}\xi_i$$
$$\text{subject to } y_i(w^T\phi(x_i)+b) \geq 1- \xi_i\ ,\ \xi_i \geq 0. \tag{3}$$

Where, function $\phi$ map training vector $x_i$ into a higher dimension space and C is the capacity constant. The value of C is to avoid overfitting because if the C is large, the error is large penalized. $K(x_i, x_j) \equiv \phi(x_i)^T\phi(x_j)$ is called kernel function. There are many kernel functions such as linear, polynomial, RBF (Radial Basis Function) and sigmoid. The RBF kernel is used most widely and equation is given by.

$$K(x_i, x_j)=\exp(-\gamma \parallel x_i- x_j \parallel ^2),\ \gamma>0 \tag{4}$$

### 2.3.4. Bernoulli Naïve Bayes

Naïve Bayes method is probabilistic algorithm based on Bayes' theorem which is stated mathematically in the following equation.

$$P(A|B)= \frac{P(B \mid A)\, P(A)}{P(B)} \tag{5}$$

Where, A and B are events and P(B) ≠ 0. Each P(A) and P(B) are the probabilities of observing A and B independently of each other. P(B|A) is a conditional probability. In classification task, P(A|B) is the probability of given data B belongs to class A. it is called the likelihood
In this paper, we used Bernoulli Naïve Bayes which required binary feature vector. The decision rule for BNB is based on the following equation [16].

$$P(x_i \mid y)=P(i \mid y)x_i+(1-P(i \mid y))(1-x_i) \tag{6}$$

# 3. Design of Experiment

In this paper, we studied performance of N-gram feature vector by the several different value of N. The purpose of experiment is evaluating the performance of feature extraction methods according to N-gram. The overall flow chart of experiment is shown in fig. 2.


**Fig. 2:** The overall flowchart of experiment.

In this paper, we used total 1224 PE files as dataset. The benign is 445 files that was extracted from Window file directory. The malware is 779 files got from the Web sites, “Virusshare [17]”, “malwareurls jpxeankoret [18]”, “malc0de [19]” and “malware-blacklist [20]”. The malware files are consisted with 418 Trojan, 176 PUP, 58 Virus, 34 Backdoor, 29 Adware, 21 Downloader, 13 Spyware and so on.

In the preprocessing step, we analyzed the execution file of the PE file form by using pefile module and disassembled it with capstone module. After disassembling, we got instruction ID, address of the instruction, mnemonic of the instruction, operands of the instruction, the size of instruction and the byte sequence of the instruction. So, if we use two modules we can extract Opcode or API functions from them to make a feature vectors.

In this experiment, we extract feature vectors by using the same methods which are used in the paper [9]. Then, we made feature vector with N-gram technique additionally. The feature vectors are constructed with Boolean values. Each Boolean values represent whether each N-gram sequence is existed (1) or not (0). We used 1-gram, 2-gram and 3-gram according to the size of N.

Next, we used several classifiers to measure the performance of them. These classifiers are DT, KNN, SVM and BNB offered by Scikit-learn in python. In Bernoulli Naïve Bayes Classifier, we use 0.5 for alpha value which is smoothing parameter. In Decision Tree Classifier, to make same tree structure in every repetition, we set random state for 0 and use entropy for impurity measurement method. In Support Vector Classifier, we use default parameter value. In K-Nearest Neighbor Classifier, we use the value of K as 3 for number of neighbor value.

## 4. Experimental Result

To measure the performance of malware detection according to the size of N-gram, we used performance of classifiers. If we measure the performance of the same classifier with different size of N-gram, it means that we measure the performance of N-gram indirectly. There are many available methods to measure the performance of classifier. These are accuracy, confusion matrix, precision and recall, ROC (Receiver Operating Characteristic) curve with binary classifiers and so on.

In this paper, we used the accuracy and the RCO curve. At first, we selected methods of feature extraction by measuring accuracy for malware detection. After that, we measure the ROC curve which is TPR (True Positive Rate) against the FPR (False Positive Rate). The FPR is the ratio of negative instanced that are incorrectly classified as positive. One way to compare the performance of classifier is to use AUC (Area Under the Curve) score of ROC curve. The perfect classifier will have AUC score equal to 1, whereas a purely random classifier will have AUC score equal to 0.5. We used four classifiers for our experiment. These are DT (Decision Tree), SVM (Support Vector Machine), KNN (K-Nearest Neighbor) and BNB (Bernoulli Naïve Bayes) classifier.

We measured the accuracy of malware detection according to the size of N-gram by using several classifiers to select feature extraction methods. The results are shown in table 1 and fig. 3.

**Table 1:** The accuracy according to the three methods

| Classifier | N | Feature: Opcode Only | Feature: Both Opcode & API | Feature: API Only |
|---|---|---|---|---|
| DT | 1 | 94.02 | 95.38 | 93.48 |
| | 2 | 97.83 | 97.55 | 82.88 |
| | 3 | 97.28 | 97.28 | 81.52 |
| KNN | 1 | 95.65 | 95.11 | 92.66 |
| | 2 | 97.28 | 97.28 | 55.71 |
| | 3 | 97.83 | 97.83 | 54.62 |
| SVM | 1 | 92.93 | 92.93 | 78.53 |
| | 2 | 93.75 | 93.48 | 66.30 |
| | 3 | 93.75 | 93.75 | 66.30 |
| BNB | 1 | 83.97 | 86.14 | 93.48 |
| | 2 | 80.98 | 84.51 | 82.34 |
| | 3 | 78.80 | 78.80 | 80.16 |

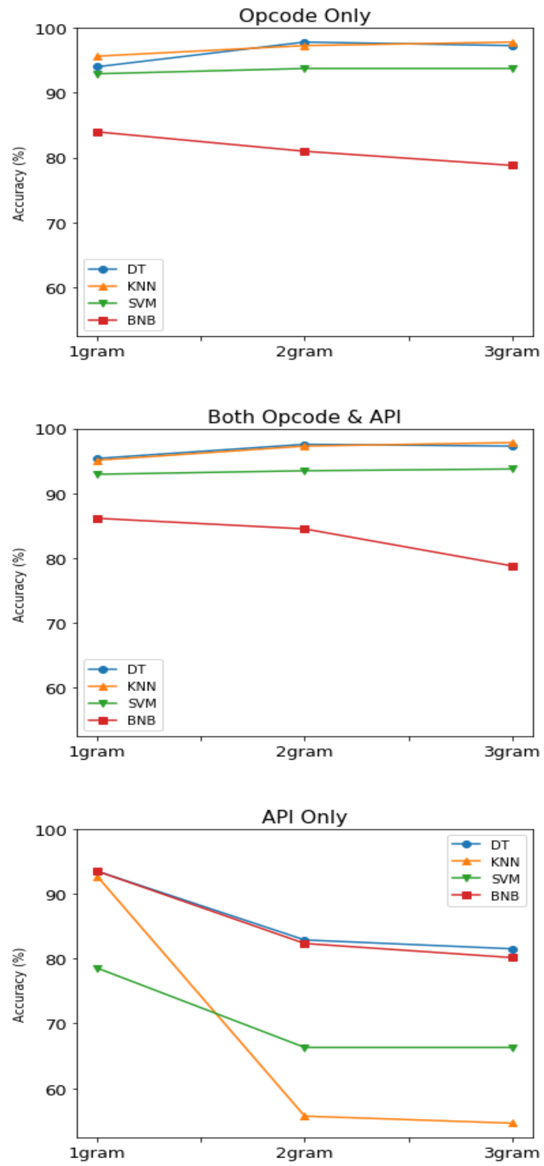



**Fig. 3:** The graph of accuracy according to the three methods

In table 1 and fig. 3, we can see the accuracy scores according to the size of N-gram for three feature extraction methods, those are Opcode Only, both Opcode and API, API Only.

In the case of Opcode Only, the accuracy of DT and KNN are better than other classifiers according to the size of N-gram. It shows that the accuracy gets higher for every classifier except BNB when the size of N-gram is increased. In the case of Both Opcode and API, it shows same result with Opcode Only.

In the case of API Only, the accuracy of DT and BNB are better than other classifiers. Also, it shows that the accuracy gets lower for every classifier when the size of N-gram is increased. This is contrast with other two extraction methods.

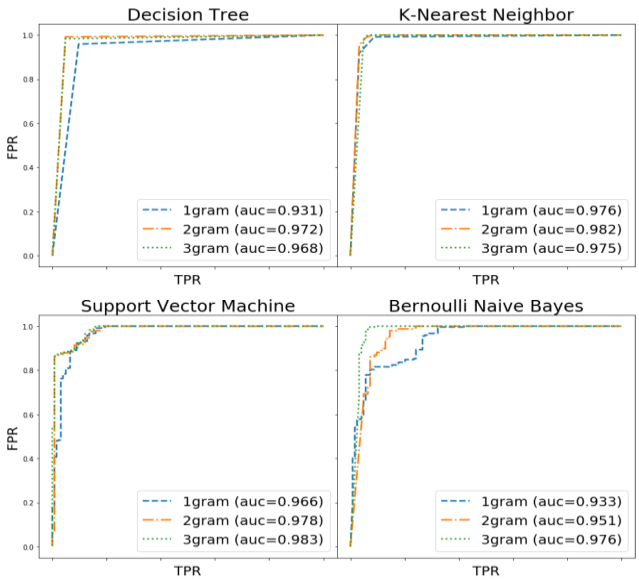
In the case of all with BNB classifier, it got low accuracy when the size of N-gram is increased. We can guess the reason that the classification performance of BNB classifier performs well with small vocabulary sizes, but its performance gets lower when the vocabulary size is large because of large input dimension in text classification task [21]. Therefore, we found the low performance of BNB classifier because input dimension of classifier is also increased when the size of N-gram is increased.

As the result of experiments, we found Opcode have more distinct characteristic by itself and its sequences than Window API calls

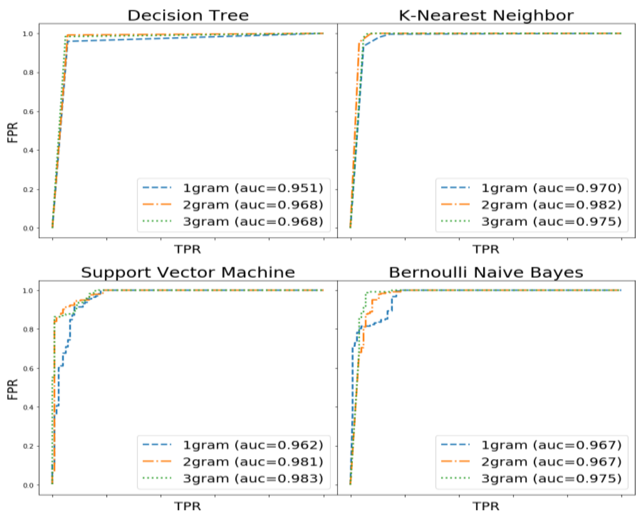
for malware detection. Therefore, we can conclude that feature extraction methods by using Opcode Only and both Opcode and API are reasonable for malware detection.
Next, we measured AUC score of ROC curve because accuracy can influence by the way of composing of dataset. If dataset is imbalanced, the classifiers can have good accuracy on the majority class but poor on the minority class [22]. Therefore, we additionally measured the AUC score to compare the robustness of feature vector according to N-gram. The results are shown in table 2 and fig. 4.

**Table 2:** The AUC score according to the three methods

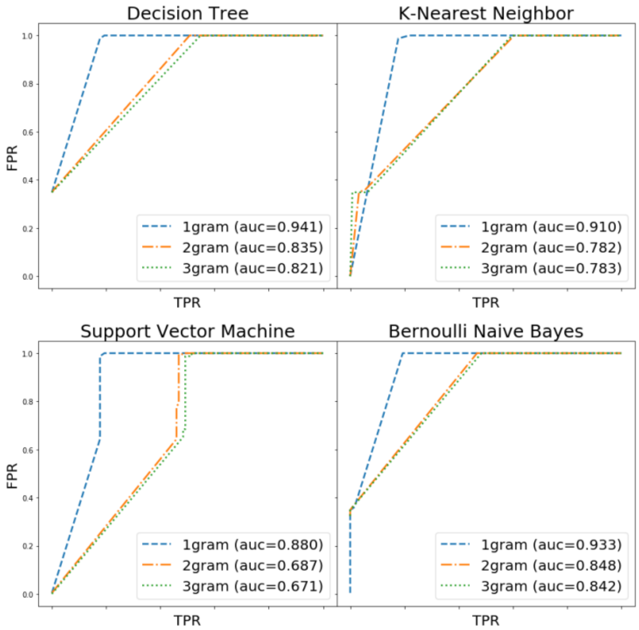
| Classifier | N | Feature | | |
|---|---|---|---|---|
| | | Opcode Only | Both Opcode & API | API Only |
| DT | 1 | 0.931 | 0.951 | 0.941 |
| | 2 | 0.972 | 0.968 | 0.835 |
| | 3 | 0.968 | 0.968 | 0.821 |
| KNN | 1 | 0.976 | 0.970 | 0.910 |
| | 2 | 0.982 | 0.982 | 0.782 |
| | 3 | 0.975 | 0.975 | 0.783 |
| SVM | 1 | 0.966 | 0.962 | 0.880 |
| | 2 | 0.978 | 0.981 | 0.687 |
| | 3 | 0.983 | 0.983 | 0.671 |
| BNB | 1 | 0.933 | 0.967 | 0.933 |
| | 2 | 0.951 | 0.967 | 0.848 |
| | 3 | 0.976 | 0.975 | 0.842 |



**(a)** Opcode Only



**(b)** Both Opcode and API



**(c)** API Only

**Fig. 4:** The graph of AUC score according to the three methods.

In table 2, we can see the AUC score of classifiers according to the size of N-gram for three feature extraction methods. In fig. 4, we can see the ROC curve and AUC score that Opcode Only at (a), Both Opcode and API at (b) and API Only at (c).
In the case of Opcode Only, the KNN and SVM are better than other two classifiers because the AUC score of them are high and the variance is low. Also, it shows that AUC score is increased in every classifier when the size of N-gram is increased. In the case of Both Opcode and API, we can see the similar result with Opcode Only.
In the case of API Only, (As we mentioned earlier, feature extraction method with API Only is not works well against other two feature extraction method for malware detection, however, we did the experiment for assurance) the AUC score of DT and SVM are better than other classifiers. Also, it shows that AUC score got lower when the size of N-gram is increased.
Through our experiments, we discover some facts. First, we recommend Opcode Only method because it support sufficient performance and is simple. But we do not recommend the feature extraction method that API Only because it do not support sufficient performance. Second, the instance-based classifier (KNN) has good performance in the view of AUC score and accuracy if we consider the result of Opcode Only. Third, DT has good performance in the view of AUC score and accuracy among the model based classifiers (DT, SVM, BNB).

## 5. Conclusion

In this paper, we studied the performance of feature extraction methods according to the size of N-gram. We extracted feature vector in three methods, Opcode Only, Both Opcode and API, API Only. We additionally utilized N-gram technique which is frequently used in Natural Language Processing domain for malware detection. Then, we measured their performance indirectly with measuring the AUC score and accuracy of classifier.
If we use N-gram technique, we recommend Opcode Only method through our experiments. Second, the instance-based classifier KNN and DT among the model based classifier have good performance.

## Acknowledgement

This research was supported by R.O.K. National Research Foundation under grant NRF-2017R1D1A1B03036372. We thanks our colleagues from laboratory who provided insight and expertise that greatly assisted the research. And thanks to National Research Foundation who supported laboratory.


## References


[1] The Independent IT-Security Institute, https://www.av-test.org/en/statistics/malware/

[2] Ashwini Mujumdar, Gayatri Masiwal, Dr.B. B. Meshram, "Analysis of Signature-Based and Behavior-Based Anti-Malware Approaches," IJARCET, Volume 2, Issue 6, June 2013

[3] James Scott, "Signature Based Malware Detection is Dead," Institute for Critical Infrastructure Technology, February 2017

[4] Kateryna Chumachenko, "Machine Learning Methods for Malware Detection and Classification," kaakkois-Suomen ammattikorkeakoulu, March 2017

[5] Edward Raff, Jared Sylvester, Charles Nicholas, "Learning the PE Header, Malware Detection with Minimal Domain Knowledge," Proceeding of the 10th ACM Workshop on Artificial Intelligence and Security, November 2017, pp: 121-132

[6] Yibin Liao, "PE-Header-Based Malware Study and Detection," University of Georgia, 2012

[7] Igor Santos, Felix Brezo, Javier Nieves, Yoseba K. Penya, Borja Sanz, Carlos Laorden and Pablo G. Bringas, "Idea: Opcode-sequence-based Malware Detection," Engineering Secure Software and Systems, vol 5965, Springer, Berlin, Heidelberg, 2010

[8] Veeramani R, Nihin Rai, "Windows API based Malware Detection and Framework Analysis," International Journal of Scientific & Engineering Research, Volume 3, Issue 3, March, 2012

[9] Tae-Hyun Ahn, Sang-Jin Oh, Young-Man Kwon, "Malware Detection Method using Opcode and windows API Calls," The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vo1.17, No.6, December 2017, pp: 11-17

[10] Willeam B.Carnar, John M.Trenkle, "N-Gram-Based Text Categorization," In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, 1994, pp: 161-175

[11] Payal B.Awachate, Prof.Vivek P. Kshirsagar, "Improved Twitter Sentiment Analysis Using N Gram Feature Selection and Combinations," IJRCCE, Vol.5, Issue 9, September 2016

[12] Mikhail Zolotukhin, Timo Hamalainen, "Detection of Zero-day Malware Based on the Analysis of Op-code Sequences," The 11TH Annual IEEE CCNC – Security, Privacy and Content Protection, 2014

[13] Scikit Learn, http://scikit-learn.org/stable/modules/tree.html

[14] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'REILLY, 2017, pp: 167-179

[15] Chih-Wei Hsu, Chin-Chung Chang, Chih-Jen Lin, "A Practical Guide to Support Vector Classification," 2016

[16] Scikit Learn, http://scikit-learn.org/stable/modules/naive_ bayes .html

[17] virusshare, https://virusshare.com

[18] joxeankoret, https://malwareurls.joxeankoret.com

[19] malc0de, https://malc0de.com

[20] malwareblacklist, https://www.malwareblacklist.com

[21] Andrew McCallum, Kamal Nigam, "A comparison of Event Models for Naïve Bayes Text Classification," AAAI Workshop, 1998, pp: 41-48

[22] Vaishali Ganganwar, "An overview of classification algorithms for imbalanced datasets," International Journal of Emerging Technology and Advanced Engineering, Vol 2, Issue 4, April 2012