ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

N-Grams and Neural Networks in Early Virus Warning

Mohamed H. Almeer

Computer Science & Engineering Department, Qatar University Doha, Qatar

Abstract*:* This paper proposes an intelligent first-warning system for virus code detection based on neural learning in an artificial neural network (ANN). The system operates in accordance with the basic principles of ANNs for pattern matching, in which the detectors detect a virus signature after training by means of analysis of the byte content of the executable code. ANNs provide the potential to identify and classify network activity based on limited, incomplete, and nonlinear data. The proposed system is capable of accurately detecting virus codes learned by training, and gives false positive ratios within acceptable ranges. The results of experiments conducted indicate that the combination of N-grams and neural networks results in a low false positive rate. The key ideas and approaches necessary for adaptation and adjustments when implementing a neural network model as an underlying early warning virus detection system are also discussed.

Keywords*:* neural networks, virus recognition, N-grams, antivirus software, ClamAV

I. INTRODUCTION

The majority of antivirus software products currently commercially available utilize signature-based virus detection and heuristic classifier models that have the ability to detect new viruses. However, the 'classic' signature-based detection algorithms simply use byte signatures of known viruses saved in memory to generate detection models. In general, detection methods based on byte signatures use a huge collection of regular expressions or simple signature string-matching engines to scan files. Signatures create a unique tag for each virus which can be considered a fingerprint, so that future examples of it can be correctly identified with a small or acceptable false positive error rate. The signature-based approaches currently used in antivirus products have acceptable detection rates for known viruses in addition to low false positive and low false negative rates.

The term 'N-gram analysis' is used in language modelling and speech recognition, but character N-grams were used prior for text categorization purposes. The Common N-gram analysis method [1], for instance, is used for text classification, authorship recognition [2], and text clustering [3]. The 3-grams (tri-gram), in particular, can perform very well, although it appears to be too short to be able to memorize any noticeable information sequence. However, some practitioners use N-grams to detect features of code that are unique to certain tools, code generators, compilers, assemblers, or programming environments.

In addition, N-grams are still used to capture features that could be unique for personal coding or even the coding styles of individuals. An N-gram can be visualized as a fixed-size sliding window byte array in which 'N' is the size of that window. For example the sequence 'ABCDEFGH' is segmented (represented) in 5-gram as 'ABCDE', 'BCDEF', 'CDEFG', 'DEFGH', etc.

The byte N-gram used in this study to detect computer viruses has been studied extensively earlier in computer virology research. In the early days of antivirus software, a byte N-gram-based method was successfully used to automatically extract virus signatures and to measure similarities in real-time processes. The representation of viruses using N-gram profiles has been investigated by various researchers, with good virus detection results obtained [2], [3], [6]. The first known use of machine learning in virus detection was carried out by Tesauro et al. [4] and Arnold and Tesauro [5]. Their detection algorithm was implemented in IBM's antivirus scanner, which has been used for years to detect boot sector malware. They used 3-grams as a feature set and neural networks as a detection and classification method.

Further, Abou-Assaleh et al. [6] researched use of the N-grams-based signature method to detect computer viruses. He used a k-NN algorithm and conducted experiments that produced results with very good detection ratios. However, no false positive ratio was reported and they did not utilize a neural network. The major difficulty in using byte N-grams for classification and/or recognition is that the set of byte N-grams obtained from the byte strings of viruses, apart from benign programs, is very large. Therefore, occurrences of these signatures in benign or clean files are still very likely to appear, although it does not point to occurrences for viruses. Thus, it is more convenient to restrict utilization to a smaller set of related N-grams for recognition of the viruses and use benign executables as learning patterns. Many studies have addressed this problem, but this study details how neural network parameters affect recognition. In this study, we use the N-gram feature of the virus to recognise the virus instances with respect to their trained group (that is, if training is implemented on 500 viruses, then the indicator only triggers when one of those viruses appear in the corpus of the file currently being scanned). This study focuses only on signature detection and particularly uses byte lengths ranging from 4-grams to 18-grams in one or two (start and end sides) different places(s) from the corpus of the virus.

We also analyse the applicability of neural networks in identifying instances of a virus within the files searched, and discuss tests conducted in which neural network-based detectors were utilized as conceptual prototypes. Those tests ranged from changing the virus numbers (called Virus Code VC patterns) that must be recognised to changing the benign parts of sample files (called Benign Code BC patterns) needed for training and observing the false positive metric. In addition, changing the training error goals and the effect of changing the number of neurons in the hidden layer are also studied, with important results obtained. We therefore implemented an N-gram, augmented with neural network ability for memorization, to determine the effectiveness of such algorithms in recognising viruses. This study seeks to investigate the extent to which the use of neural networks facilitate successful identification and recognition of viruses based on N-grams signatures.

II. ARTIFICIAL NEURAL NETWORKS (ANNS)

An ANN consists of a number of interconnected processing elements and maps a set of inputs to a set of desired outputs. The characteristics of the elements and the weights associated with the interconnections among them determine the result of the transformation. The nodes of the network are able to easily adapt to the desired outputs by changing the connections between their links. An ANN analyses information and gives a probability estimate of the data matching the required pattern which it has been trained to recognise. This characteristic makes ANNs one of the most desired methods for pattern recognition and signature matching. However, the decision as to the accuracy of the matches still relies completely on the experience of the system (the memorization process) inculcated during the training phase using examples of the problem in question. Fig. 1 depicts a typical multilayer feedforward NN.

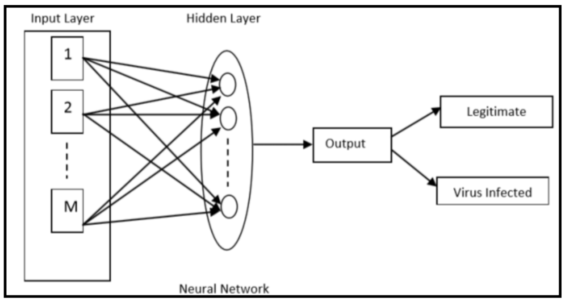

Fig. 1. Typical multilayer neural network circuit

III. DATA PREPARATION AND TRAINING

A. ClamAV Background

In this study, we utilize signatures from ClamAV [15], the most widely used open-source virus scanner. ClamAV offers client-side protection for personal computers, as well as protection for mail and file servers in large organizations. The ClamAV virus database is updated at least every four hours and, as of 25 December 2014, contained over 3,700,000 virus signatures with the daily update Virus DB number at 19,837. It consists of a core scanner library and a command-line utility.

The ClamAV database contains signatures for non-polymorphic viruses in simple string format, and for polymorphic viruses in regular expression format. The current version of ClamAV utilizes a simplified version of the Boyer-Moore algorithm [11] and simple fixed string signatures to detect non-polymorphic viruses. It uses a variant of the classical Aho-Corasick and Wu-Manber algorithms for polymorphic viruses [12], [13]. The simple format in which ClamAV stores its virus signatures, and its user-friendly conversion utility, with which it is equipped, enabled us to extract the N-gram signatures utilized in training the ANN used in our design from its databases of patterns.

Four programs, written in the C language, have been developed to assist in extraction of the required N-grams signature corpus of the virus body, pre-processing of data to make it available for training in MATLAB Neural Network Toolbox, and to prepare the patterns from collections of executable files:

1. The first program acts on the main ClamAV signature database file and converts the content from binary to hex format in ASCII. The 'main.cvd' database file in which ClamAV stores all its signatures contains the static signatures of 3.7 million viruses.
2. The second program extracts more details about the virus corpus and deletes or bypasses head identification data, such as the name of the virus.
3. The third program chooses a user-specified number of virus N-grams, in any sequence, to be further processed by the Neural Network Toolbox in MATLAB [14].
4. The fourth program reads collections of Windows executable files then arranges them in text files with sliding window effects. This preparation is essential in order for MATLAB to apply the collection of patterns to the Neural Network for validation. Nevertheless, this maximizes the size of the generated files, for instance a file such as 'adamsync.exe', with size 164 KB, will generate text files with sizes of 238 KB (for 4-grams), 343 KB (for 6-grams), 447 KB (for 8-grams), and 464 KB (for 10-grams).

IV. NEURAL NETWORKS AS PATTERN RECOGNITION MODELS

VC detection can be viewed as a binary classification problem; therefore, we can use a multilayer ANN that operates as a pattern matcher to conduct the detection process. The steps in the proposed detection procedure can be summarized as follows:

- Dump hexadecimal byte sequences from viruses taken from the 'main.cvd' ClamAV main virus database and benign installed '.exe' executable files that exist in any PC.

- Slice each Hex sequence into gram by N—the size of the sliding window—and save them in a file to be presented to the trainer.
- Implement training and validation of the model.

Tables 3 and 4 give details on the content of one of the viruses used in ClamAV and its format, as seen in the ClamAV main database file, and show the names of the 50 viruses used in this study, respectively.

In reality, it is likely impossible to collect all normal variations of a safe code. Thus, the possibility exists that our normal collection of BC will give incomplete coverage of normal behaviour. If the normal coverage is incomplete, then false positives could result. The focus of this research is on investigation of the performance of such virus detection solutions based on neural networks under incomplete training patterns. We selected random executable files containing random sequences of bytes representing the benign patterns we used in training.

The most widely used ANN, the multilayer feedforward neural network, is used in this study because it is considered the 'workhorse' of neural networks in general. It can be used for both function fitting and pattern recognition applications (as in our case). The ANN used in this study comprises an input layer, a hidden layer, and an output layer.

In order to standardize our comparison of the various virus pattern recognition approaches, we froze some properties of the ANN—specifically, the number of layers (a total of three, two hidden and one output); the number of neurons in the input layer, which is used by the N-gram for the size of the sliding window; the number of neurons in the first hidden layer, 200; the number of neurons in the second hidden layer, 10; and the number of neurons in the output layer, one (because it signals true or false according to whether the virus is present or absent, respectively). Table.1 illustrates the details.

Moreover, the activation functions used in both the input and hidden layers were chosen to be sigmoidal, whereas a linear activation function was used for the output layer. Fig. 3 depicts the ANN used in this study. It is recommended that when training large networks, and in particular recognition networks, that the training algorithms TRAINSCQ and TRAINRP be used because they are the best in such scenarios.

Their memory requirements are relatively small, and yet they are much faster than standard gradient descent algorithms. TRAINRP is the more suitable for pattern recognition, and is good enough for large networks with many neurons and large datasets, because it is the fastest algorithm known. Our network has more than 200 neurons and a large number of training sets, which justifies the use of 'TRAINRP', or a resilient backpropagation algorithm.

TABLE 1 NEURAL NETWORK CHARACTERISTICS

Property	**Value (Range)**	
Number of layers	3 (Input-Hidden: 1-Hidden, 2-Output)	
Neurons and activation function (per layer)	**Layer1**	(200) Sigmoidal
	Layer2	(10) Sigmoidal
	Layer3	(1) Linear
Window size (N-grams) Input layer size	3-4-6-8-10-12-14-16-18.	
Training algorithm	(RP) TRAINRP Resilient Back propagation.	
Training error goal	1.0E-3, 1.0E-4, 1.0E-5, 1.0E-6, 1.0E-7, 1.0E-8	
Max epochs allowed (given)	100,000 Iterations	
Computer (OS) used	Lenovo Intel® Core i3-3217U CPU, 1.8 GHz, 4 GB RAM, 64-bit OS, X64-based processor (Windows-8.1 Pro)	

V. EXPERIMENTAL EVALUATION

We utilized the main virus signature database of the ClamAV program [15]. The collection is composed of over 100,000 static virus signatures and approximately 0.75 million MD5 hashed patterns. We extracted 50, 100, 250, and 500 virus codes as VC. Then, we collected more than 125 Win32 executable files from a fresh installation of Windows 8, with applications installed, on an ordinary Intel-based PC. The files had a total size of over 54 MB. We considered this collection benign code (BC) and divided it into two sets; the first set for training, and the second set to validate the network.

In order to achieve more efficient use of the validation method, we focused our attention on the false positive rate as a performance indicator. We also specified lower and upper bounds in order to determine when a correct recognition is made. Thus, the presented patterns for the network trigger a positive recognition only when the output falls between those two bounds (Table 2). The bounds were recorded at the end of the training process. Fig. 2 shows the various ranges for the system simulated from its own training datasets.

TABLE.2 NEURAL NETWORK RECOGNITION RESULTS FOR TRAINING ERROR GOAL = 1.0E-4 AND NETWORK = 200-10-1 NEURONS.

	Bounds for Positive Recognition		**Training Epochs**	**False Positive**
Virus Number	**Min**	**max**		
50	0.7537	0.9505	636	3.9 %
100	0.7465	0.9694	1532	9.6 %
250	0.8459	0.9222	3557	9.0 %
5000	0.8229	0.9277	2772	31 %

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

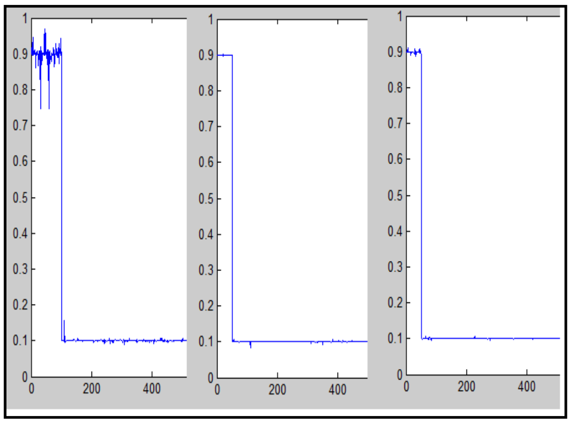

Fig. 2. Positive and negative recognition outputs attained by training when various training properties are considered.

A. *Experiment 1*

This first experiment tested the validity of the networks when the number of BC increased along with the number of virus patterns; while maintaining the VC to be recognised at 50 viruses. BC ranged from 250, 500, 1000, 1500, to 2000. Fig. 3 shows the results obtained. The x-axis shows BC as a negative indicator for the presence of viruses, while the y-axis depicts the false positive rate in percentage.

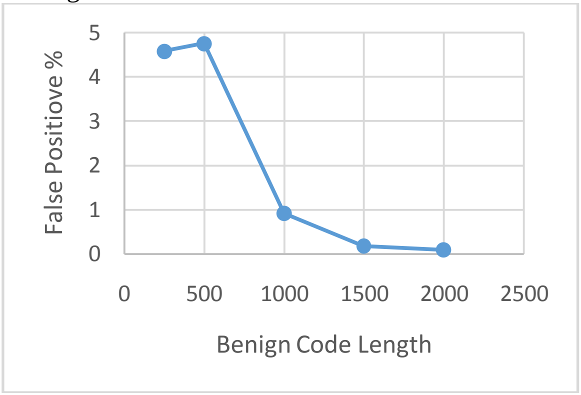

Fig. 3. Change in false positives for various BC numbers

A different experiment that tests the changing of VC versus performance was also conducted. The results are depicted in Fig. 4 and Table 2. The results are encouraging. They show an FP ratio of 0.0928% at BC length = 2000 B. A high accuracy of > 0.098% and even more can be achieved if BC > 2000 is tested. The results indicate the attainment of an FP ratio between 3.9% and 31% when the VC number increased from 50 to 500.

However, increasing VC while maintaining BC is considered a load on the ANN, which must then memorize more patterns as positive while maintaining the number memorized for negative recognition unchanged. That leads to a more complicated network, and hence, worse training and recognition results.

Therefore, increasing BC will get better results and improves the recognition as opposed to increasing VC.

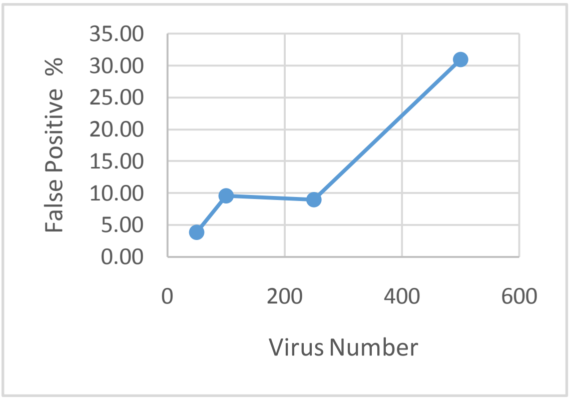

Fig. 4. Change in false positives for various VC numbers

B. *Experiment 2*

This second experiment was conducted to determine how changing the number N in N-grams affects performance. The test was conducted over the range N = 3, 4, 6, 8, 10, 12, 14, 16, to 18. Fig. 5 shows the results obtained. The x-axis indicates the number of N-grams; while the y-axis depicts the false positive ratio in percentage. The results (Fig. 5) are encouraging, because an FP ratio of 0.0004% is achieved for N = 18. High accuracy is also expected for N > 18. Thus, we conclude that increasing the N value in N-gram results in a better virus recognition measure; however, a higher calculation load is involved and hence more processing time is required.

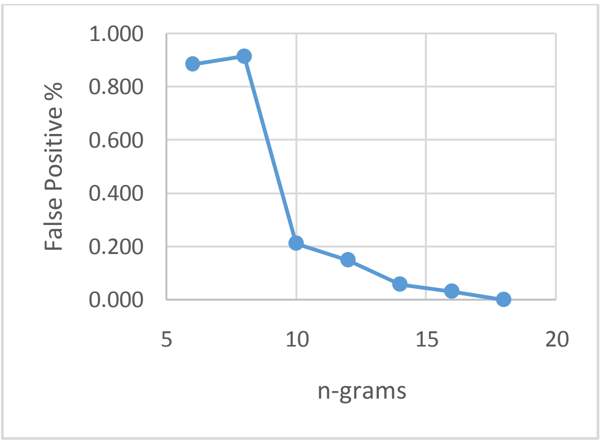

Fig. 5. Change in false positives for various N values in N-grams

C. *Experiment 3*

The third experiment validated the effect of changing the accuracy of the training over the range 1.0E-3 to 1.0E-8 in increments of 0.1. Here, the degree of overfitting was varied and the effect on the virus recognition process recorded. Overfitting occurs when the classifier learns the training set too well. Fig. 6 shows the results obtained. The x-axis indicates the training accuracy the network has converged to; while the y-axis depicts the false positive ratio in percentage. In most neural network studies, overfitting is avoided because there is no need to generalize the recognition to a wider range of inputs that has not been presented at the training phase. Conversely, in this case, overfitting is used as a measure of how close

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

the recognition will be when presented by the correct virus patterns; in addition, other non-virus patterns should be avoided.

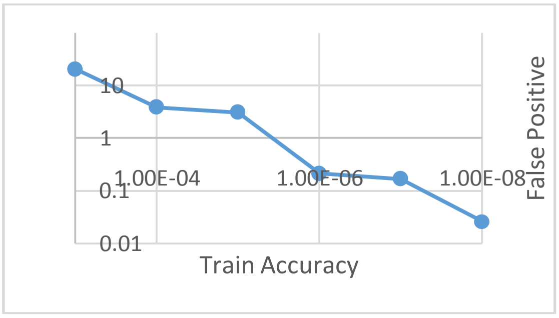

Fig. 6. Changes in false positives for various training accuracy goals

TABLE 3 VIRUS SIGNATURE FORMAT IN CLAMAV AND ITS REDUCED CONTENT USED IN THIS STUDY FOR N=16.

Virus Name	Trojan.Proxy.Ranky-45
Detailed pattern in "main.cvd"	Trojan.Proxy.Ranky-45:1:*:c6d7435624d832415cf0b5eeaf5b302f5765319e08957b8404f92f956381e4f33f56e26e9ceff8f2a8eec259dd89af336911a7564a2b0802779a3a2b5beb9bc576e3610e405823faff872cab2268b5b9e8c4401a8747941015e36680762559c8bc948b523d27ce29a700e5dbe2543f7904174250d54b3cb1633a976458d0d5070f0ce5837466603aa4eb575e7d622bf8465271ef14
Extracted and summarized pattern used in the study	c6d7435624d832415cf0b5eeaf5b302f5765319e08957b8404f92f956381e4f33f56e26e9ceff8f2a8eec259dd89af336911a7564a2b0802779a3a2b5beb9bc576e3610e405823faff872cab2268b5b9e8c4401a8747941015e36680762559c8bc948b523d27ce29a700e5dbe2543f7904174250d54b3cb1633a976458d0d5070f0ce5837466603aa4eb575e7d622bf8465271ef14
N-grams = 16 Bytes	c6d7435624d832415cf0b5eeaf5b302f

VI. CONCLUSION

In this study, we presented encouraging preliminary results obtained in applying a feed forward neural network based on byte the N-gram format to the detection of virus codes using virus patterns extracted from ClamAV, a popular open-source free antivirus program. The method achieved a false positive rate of 0.0004% for N-gram>18 on training data, 0.09% for BC = 2000, and FP ratio of 0.026% for a training error goal of 1.0E-8. In future work, we plan to conduct experiments on larger data collections, mining the extracted N-grams to refine the method for extraction of the VC signatures, and consider the implementation of this method on FPGA or Reconfigurable Logic.

TABLE 4 FIRST 50 VIRUSES FOUND IN CLAMAV AND USED IN THIS STUDY

No.	**Virus Name**	**No.**	**Virus Name**
1	Exploit.HTML.ObjectType	26	HTML.Phishing.Bank-27
2	Exploit.HTML.DragDrop	27	HTML.Phishing.Bank-23
3	HTML.Phishing.Bank-4	28	HTML.Phishing.Bank-25
4	W32.MyLife.E	29	HTML.Phishing.Bank-26
5	Trojan.Downloader.Small-140	30	Trojan.Banker-7
6	HTML.Phishing.Auction-1	31	HTML.Phishing.Bank-29
7	HTML.Phishing.Auction-2	32	HTML.Mydoom.email-gen-
8	HTML.Phishing.Bank-5	33	Trojan.Spy.Flux.A-srv
9	HTML.Phishing.Bank-6	34	Trojan.Downloader.Small-
10	W32.Wabrex.A	35	HTML.Mydoom.email-gen-
11	HTML.Phishing.Bank-8	36	HTML.Mydoom.email-gen-
12	HTML.Phishing.Bank-9	37	HTML.Mydoom.AD-
13	HTML.Phishing.Bank-11	38	HTML.Phishing.Bank-32
14	HTML.Phishing.Bank-12	39	HTML.Phishing.Bank-31
15	Trojan.Downloader.VBS.Psyme.H	40	HTML.Phishing.Bank-34
16	Trojan.Flashkiller.A	41	HTML.Phishing.Bank-35
17	HTML.Phishing.Bank-13	42	Exploit.HTML.IFrameBOF-
18	HTML.Phishing.Bank-16	43	HTML.Phishing.Auction-3
19	HTML.Phishing.Bank-15	44	Trojan.Downloader.Psyme-
20	Trojan.Downloader.JS.Psyme.T	45	Trojan.Downloader.Monurl-
21	Trojan.HacDef-1	46	HTML.Phishing.Bank-38
22	Trojan.Regger-1	47	HTML.Phishing.Bank-37
23	HTML.Phishing.Bank-18	48	HTML.Phishing.Bank-36
24	HTML.Phishing.Bank-20	49	HTML.Phishing.Auction-4
25	HTML.Phishing.Bank-21	50	Trojan.Downloader.Small-

REFERENCES

[1]. D. Reddy, K. Sandeep, and A. K. Pujari. "N-gram analysis for computer virus detection," Journal in Computer Virology, vol. 2, no. 3, pp. 231–239, 2006.
[2]. W. Cavnar and J. Trenkle, "N-gram-based text categorization," in Proc. SDAIR-94, 1994.
[3]. G. Frantzeskou, et al. "Effective identification of source code authors using byte-level information," in Proc. 28th International Conference on Software engineering. ACM, 2006.
[4]. G. J. Tesauro, O. J. Kephart, and B. G. Sorkin, "Neural networks for computer virus recognition," IEEE EXPERT Magazine, pp. 5–6, 1996.
[5]. W. Arnold and G. Tesauro, "Automatically generated Win32 heuristic virus detection," in Proc. Virus Bulletin Conference, pp. 51–60, September 2000.
[6]. T. Abou-Assaleh et al., "N-gram-based detection of new malicious code," in Proc. COMPSAC 2004, vol. 2, 2004.
[7]. K. Tan, "The application of neural networks to UNIX computer security," in Proc. IEEE International Conference on Neural Networks, vol. 1, 1995.
[8]. S. Shah, H. Jani, S. Shetty, and K. Bhowmick, "Virus Detection using Artificial Neural Networks," International Journal of Computer Applications, vol. 84, 2013.
[9]. I. Santos, Y. K. Penya, J. Devesa, and P. Garcia Bringas, "N-grams-based File Signatures for Malware Detection," ICEIS, vol. 2, no. 9, pp. 317-320, 2009.
[10]. B. Zhang, et al., "New malicious code detection based on n-gram analysis and rough set theory," in Proc. International Conference on Computational Intelligence and Security, vol. 2, 2006.
[11]. R. S. Boyer and J. S. Moore, "A fast string searching algorithm," Communications of the ACM, vol. 20, no. 10, pp. 762-772, 1977.
[12]. A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," Communications of the ACM, vol. 18, no. 6, pp. 333-340, 1975.
[13]. S. Wu and U. Manber, "A fast algorithm for multi-pattern searching." Technical Report TR-94-17. University of Arizona, 1994.
[14]. Neural Network Toolbox User Guide,
[15]. ClamAV, www.clamav.org.