Evaluation of Random Projection for Malware Classification

Stanislav Ponomarev, Jan Durand, Nathan Wallace, Travis Atkison
Louisiana Tech University
Ruston, LA, 71270
{spo013, jrd037, nsw004, atkison}@latech.edu

***Abstract*—Research efforts to develop malicious application detection algorithms have been a priority ever since the discovery of the first "viruses". Various methods are used to search and identify these malicious applications. One such method, *n*-gram analysis, can be implemented to extract features from binary files. These features are then be used by machine learning algorithms to classify them as malicious or benign. However, the resulting high dimensionality of the features makes accurate detection in some cases impossible. This is known as "the curse of dimensionality". To counteract this effect, a feature reduction technique known as *randomized projection* was implemented. Through this reduction, not only are classification times decreased but also an increase in true positive and decreases false positive rates are observed. By varying the *n*-gram size and target feature size it is possible to fine-tune the accuracy of machine learning algorithms to reach an average accuracy of 99%.**

***Index Terms*—Computer security, Data mining, Feature extraction**

I. Introduction

Since the discovery of the "Elk Cloner" computer virus in 1981 [1], the rate of development and infection of malicious software applications has increased exponentially. As the internet grew, it became easy for many types of malicious software or "malware" to gain fast and reliable access to victims' machines. Malware is a catch-all phrase used to refer to any program that is designed to "harm or subvert a system's intended functionality" [2] and falls under several categories, mainly viruses, worms, and Trojans. In practice, consumers and companies commonly refer to any malicious application, despite their idiosyncrasies, as a virus. In defense, software tools such as anti-viruses and intrusion detection systems have been developed by the cyber security community to detect and disable attacks from such malicious applications [3].

Traditionally, these defense tools used signature-based detection. This involves comparing the byte sequence of a suspected malicious application to the signatures of known threats stored in a virus signature database. If part of the application matches a signature in the database then the application in question is flagged as malicious. Signature-based detection is "effective when the virus code does not change significantly over time" [4]. In fact, a single virus signature can be used to match multiple variants of a particular virus if they all contain the base signature code [4]. Signature-based detection is a very practical method of malware detection and became popular due to its ease of use and low false-positive rates [5]. However, this approach is fundamentally limited to detecting only known threats. That is, signature-based detection tools are incapable of detecting new and previously unknown threats which do not contain any known signatures. This limitation leaves signature-based anti-virus tools completely ineffective against "zero-day" viruses until their signature database, aka virus definitions, have been updated with the signatures of the new threats. As a result, signature-based detection tools require frequent updates of the signature database to keep up with new malware [3]. This process can be very time consuming as suspicious applications must first be identified before they can be analyzed. This usually involves waiting until such applications have been reported for attacking several systems or networks. Once the suspect malicious application has been procured, its malicious intent must be confirmed, and then a signature i.e. a unique sequence of bytes uncommon to most other programs must be identified by an analyst. This signature is then added to the signature database to be used by the virus detection engine for matching. In addition, these databases have the potential to grow very large and cumbrous, which can hinder the detection process.

An alarming observation is that viruses are becoming easier to produce. Due to the introduction of attack kits, any cyber criminal can get these ready-made virus fabrication tools off the black market and start creating and deploying new malware into the wild [6]. According to the Symantec 2012 Internet Threat Report, such attack kits contributed over 61% (3 billion) of malware detected by Symantec in 2011 [6]. This surge of new malicious authors and malware has increased the need for research into the field of malware detection to ensure that the cyber security community has the upper hand. Christodorescu aptly describes it as a "game between malicious code writers and researchers working on malicious code detection." [4]

There have been research efforts which use other methods of detection to mitigate some of the difficulties inherent in dealing with new and unknown threats. Some of these approaches are based in the realms of information retrieval and data mining such as [7]–[10] and have yielded some promising results with respect to identifying whether a program is malicious. However, many of these research efforts are faced with the "curse of dimensionality", which refers to the challenges of computing in a high-dimensional space [11]. Dimensionality reduction techniques such as principal component analysis, latent semantic indexing and random projection allow the effects of the "curse of dimensionality" to be mitigated by moving from a high dimensional space to a lower-dimensional space. The work done in this paper contributes to the solution of malware detection by focusing on the application and effect of dimensionality reduction in the malicious software detection process. Previous experiments have shown that the use of random projection as a feature extraction and dimensionality reduction technique in the context of malware detection has encouraging results [3], [12]–[15]. It uses application feature sets reduced using the random projection technique in addition to features reduced using mutual information. The efficiency of using different parameters for random projection reduced data sets as well as *n*-gram size variation to use with different classification algorithms was tested and recorded in the results section.

II. Background

The problem of malicious application detection is very popular and well-studied, and has gathered a significant body of research. Essentially, all the different research ventures can be categorized as either static analysis or dynamic analysis. Static analysis refers to the process of determining whether an application is malicious without actually running the program in question. Dynamic analysis describes the process of determining whether a program is malicious by monitoring the behavior of a suspect program by executing it, usually within a virtual environment. Neither one of these approaches is a complete solution in itself, but each has a part to play in producing better malware detection systems.

A. *n-gram Analysis*

When dealing with information retrieval or data mining, the features extracted from the data set play a pivotal role in the success of the prediction process. The information retrieval technique of *n*-gram analysis has proven to be a valuable tool for feature extraction in several research efforts which focus on the detection and/or classification of malicious applications [7]–[9], [12]–[22]. An *n*-gram is any substring of length *n* [23]. Since *n*-grams overlap, they do not just capture statistics about sub-strings of length *n*, but also implicitly capture frequencies of longer sub-strings [17]. However, due to the high dimensionality of *n*-gram feature sets, the gathered data is a subject to the "curse of dimensionality." [11] Many of these research efforts use some form of dimensionality reduction to curb these large feature sets in order to mitigate the effects.

B. *Mutual Information*

Due to the enormity of the set of *n*-grams extracted from a program, Kolter introduced the dimensionality reduction technique of *mutual information* as a processing step to reduce the dimensionality of the program feature vectors. The most relevant of these *n*-grams were selected using the average mutual information measure from Yang et al. [24] i.e. the Information Gain calculated as:

$$IG(j) = \sum_{v_j \in \{1,0\}} \sum_{C_i} P(v_j, C_i) log \frac{P(v_j, C_i)}{P(v_j)P(C_i)}$$

where C_i is the i^{th} class, v_j is the value of the j^{th} attribute, $P(v_j, C_i)$ is the proportion that the j^{th} attribute has the value v_j in the class C_i, $P(v_j)$ is the proportion that the j^{th} *n*-gram takes the value v_j in the training data, and $P(C_i)$ is the proportion of the training data belonging to the class C_i [8]. The *n*-grams with the highest information gain were selected as the most relevant. These selected *n*-grams were then used to create Boolean vectors similar to those created by Shultz [25], with each vector element indicating either the presence or absence of a particular *n*-gram in/from the program being represented.

The Malware Collection Booster (McBoost) developed by Perdisci et al. [26] followed an approach similar to Kolter [8], but also introduced the ability to detect whether an executable was packed/compressed, and if so, could then unpack the executable before handing it over for classification. McBoost consisted of three modules: A) a classifier which determines if an executable is packed; B) a universal unpacker based on dynamic analysis; C) and a classifier which detected if an executable was malicious or benign using a bagged-decision-tree (BDT). Module C consisted of two specialized classifiers trained to distinguish between malicious vs. benign non-packed or hidden code; hidden code was the result of unpacking a packed executable.

Based on results from a comparison between an implementation of Kolter's method [8] and McBoost, Perdisci et al. concluded that Kolter's method was "biased towards detecting packed executables as malware and non-packed executables as benign, regardless of the nature of the hidden or non-packed code." [22]

C. *Random Projection*

Though the feature selection technique of mutual information has been very popular in reducing the feature sets of related research efforts, another dimensionality reduction technique which has been recently applied to the field of malware detection is random projection. Unlike the mutual information method used by Kolter [8], random projection is a feature extraction technique which embeds a high dimensional feature set into a "low-dimensional subspace using a random matrix whose columns have unit length" [27], thus creating a completely new set of features.

Random projection feature extraction technique was first introduced to the realm of malicious application detection in [12]–[15]. In [14], a vector space model was used with *n*-gram analysis to produce weighted feature vectors from binary executables, similarly to Kolter [8]. Every dimension of these vectors represented a unique *n*-gram which could be extracted from the corresponding executable. These feature vectors were then used as input to random projection algorithms in order to produce feature vectors of a reduced dimension. Three methods for random projection were used to reduce the feature vectors: 1) matrix multiplication with a random matrix of unit vectors with elements generated from a Gaussian distribution with a mean of 0 and standard deviation of 1; 2) Achlioptas' matrix multiplication with a random matrix of values of 0, +1, or -1 following a probability distribution of 2/3, 1/6 and 1/6 respectively [28]; 3) and random set projection based on the Linial-London-Rabinovich algorithm [29], which is an extension of the Johnson-Lindenstrauss [30] and Bourgain [31] algorithms.

To test the efficacy of using random projection in this particular context of malware detection in [3], [13], [14], *n*-gram feature vectors with *n*-grams of length 3, 5, and 7 were extracted from a data set of 1544 Windows formatted binary executables: 709 benign files and 835 malicious files. Different corpuses of reduced feature vectors were created using each of the different random projection techniques mentioned above, each containing feature sets of 500, 1000, and 1500 features.

A popular instance-based learning algorithm was used to classify the documents in each of the corpuses. Each document feature vector was compared to every other feature vector and classified based on the classes of the most similar vectors in the corpus. The cosine similarity measure was used to determine the similarity between feature vectors over the range of threshold values from 0 to 1.0 in increments of 0.05. Cosine similarity "has the nice property that it is 1.0 for identical vectors and 0.0 for orthogonal vectors." [32] The following formula was used for computing the cosine similarity between a query Q and a document D, where $w_{Q,i}$ is the weight of the i^{th} *n*-gram in the query and $w_{D,i}$ is the weight of the i^{th} *n*-gram in the document:

$$CosineSimilarity(Q,D) = \frac{\sum_i w_{Q,i} w_{D,i}}{\sqrt{\sum_i w_{Q,i}^2}\sqrt{\sum_i w_{D,i}^2}}$$

The experimental results were very promising and produced true positive rates for prediction as high as 0.95 and false positive rates as low as 0.02 [14], comparable to results of previous research efforts using the reduction technique of mutual information.

III. Experiment

A. Data Set

The data set that was compiled for the experiments described in this section consisted of 1622 Windows formatted binary executable files. None of the files in the data set were larger than 950 KB. Of these files, 303 were extracted from a fresh installation of the Windows XP operating system, another 406 were extracted from a fresh installation of the Windows Vista operating system, and another 78 were extracted from a fresh installation of the Windows 7 operating system. All of these sets were obtained by installing the respective operating system in a virtual environment that was installed on a commodity PC. These virtual environments were not connected to the Internet and therefore provided a safe location. This ensured that it would allow for application extraction without the worry of malicious infiltration during the gathering phase of the research effort. This process provided a total of 787 files that were in the data set and that were considered benign. The remaining 835 files for the data set were malicious Trojan horse applications that were downloaded from various websites on the Internet including http://www.trojanfrance.com and http://vx.netlux.org. These results were produced from experiments in which 7 classifiers were trained and tested under varying *n*-gram and feature set sizes, totaling 36 experiments in all. [3]

B. Methodology

Following the methodologies of Kolter [8], a vector space model was used to describe the executables in the data set. *n*-gram analysis was used to create feature vectors from the executables in the data set. A binary feature weighting scheme was used for this effort, whereby each unique *n*-gram was considered a feature, and feature vectors were created for each document in the data set by assigning a '1' to a vector dimension attribute if the corresponding *n*-gram was present in the executable or '0' if it was not. These feature vectors were labeled with their corresponding class of either malicious or benign. In performance testing, the malicious class was considered the positive class since the goal of detection was to identify malicious instances, while the benign class was considered the negative class. The binary feature weighting scheme was used as opposed to other schemes like the popular term-frequency inverse-document-frequency (TF-IDF) vector space weighting scheme because the mutual information feature selection technique is defined using a binary feature weighting method. The TF-IDF weight of a feature increases proportionally to the number of times that feature appears in a document, but is inversely proportional to the number of times that feature appears in the corpus, which helps to control the weights of features that are generally more common than others.

Six different *n*-gram sizes ranging from 2 to 7 were used in the *n*-gram analysis phase to produce multiple feature vectors representing the same executable file. Multiple *n*-gram sizes were used in order to analyze the effect of *n*-gram size on the classification of an executable feature vector. Each *n*-gram size created a set of feature vectors with a different number of features which grew exponentially with the *n*-gram size. *n*-grams of length 2, 3, 4, 5, 6, and 7 created feature vectors with dimensions of 65536, 16085252, 25368317, 47616980, 65500194, and 99057173 respectively. That is, *n*-gram analysis with an *n*-gram size of 7 yielded upwards of 99 million unique *n*-grams.

For each set, three reduced versions of each feature vector were created by reducing each vector to a feature set size of 500, 1000, and 1500 using the respective dimensionality reduction technique. This was done in order to analyze the effect of the number of features on the classification of an executable feature vector. The mutual information set was reduced by selecting the most relevant *n*-gram features based on the information gain measure from Yang et al. [24] The random projection set was first reduced to 200,000 features via mutual information before being further reduced using the method of random projection proposed by Achlioptas [28]. The mutual information preprocessing phase was used to remove the influence of less significant features and also speed up the overall reduction process.

Several machine learning algorithms were applied to the feature set, using the Waikato Environment for Knowledge Analysis (WEKA) [33]. These classification algorithms were the k-Nearest-Neighbors Instance-based learner (IBk), a Naïve Bayes classifier, a support vector machine (SVM), a decision tree (J48), and "boosted" versions of the last three classifiers [8]. These classifiers were trained and tested using stratified 10-fold cross-validation, with *n*-grams of length 4 and feature vectors of the top 500 *n*-grams. That is, the data set was separated into ten disjoint sets of the same size and one set was used as the testing set while the other nine combined were used to train the classifier. This process was conducted ten times using each subset as the test set only once, and then the results from the different runs were averaged.

Previous results show that SVM had the highest accuracy gain from the use of random projection, the accuracy raised from 96.49% to 98.15% due to increase in true positive rates and decrease in false positive rates. [3] To further improve the accuracy, *n*-gram size as well as generated feature size were varied. As described below, the best combination of parameters emerged for even better results.

IV. Results

All of the experiments were conducted on a commodity Acer PC with an AMD Phenom II triple-core processor and 4GB of DDR3 RAM, running the Ubuntu Linux operating system.

A. *n-gram Size Variation*

While conducting the experiments described above, the *n*-gram size parameter was varied in the range of 2 to 7 in order to find the effect, if any, of *n*-gram size on the accuracy of the various classifiers. For this set of experiments, feature size was maintained at 1000 features.

For the mutual information and random projection trained classifiers, there is a trend when observing the total average accuracy (average of all the average accuracies) of the classifiers versus *n*-gram size. In Figure 1 and Figure 2, the total average accuracy peaks around the mid-range of *n*-gram sizes and tapers off near the ends at n = 2 and n = 7. Figure 3 and Figure 4 illustrate the average accuracies of each classifier with respect to *n*-gram size.

The average accuracy graphs (Figure 1 and Figure 2) show that mutual information algorithm reaches higher overall accuracy, however, that is due to the fact that Naïve Bayes classifier performed much worse when using random projection (Figure 4).

It is apparent that using features with an *n*-gram size of 2 or 7 generally yielded less accurate classifiers. This is because features of *n*-gram size 2 do not provide enough

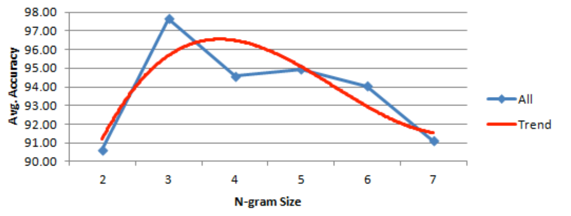

Fig. 1. Total average accuracy of all mutual information classifiers vs. *n*-gram size at 1000 features

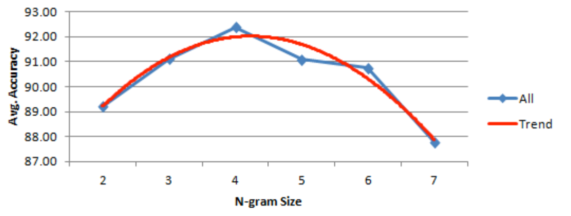

Fig. 2. Total average accuracy of all random projection classifiers vs. *n*-gram size at 1000 features

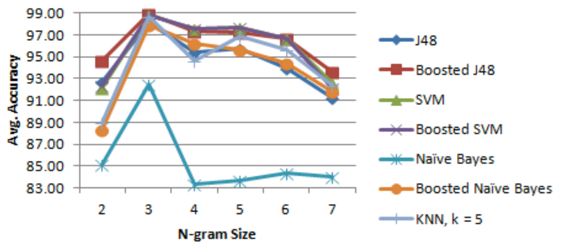

Fig. 3. Average accuracy of each mutual information classifier vs. *n*-gram size at 1000 features

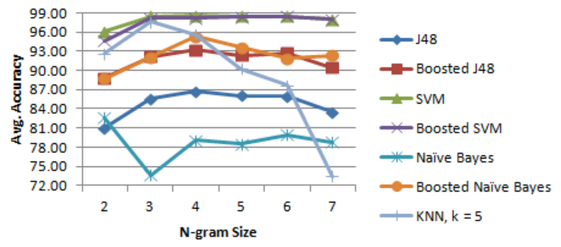

Fig. 4. Average accuracy of each random projection classifier vs. *n*-gram size at 1000 features

important information that is present in features of a larger *n*-gram size. Features of *n*-gram size 7 provide some extraneous information which impairs the classification process. Once exception though is the naïve Bayes classifier trained with random projection data which, on average, gave its most accurate predictions when dealing with an *n*-gram size of 2.

On average the classifiers were most accurate when working with *n*-grams of size 3 or 4, particularly *n*-grams of size 3 for the mutual information trained classifiers and *n*-grams of length 4 for random projection trained classifiers.

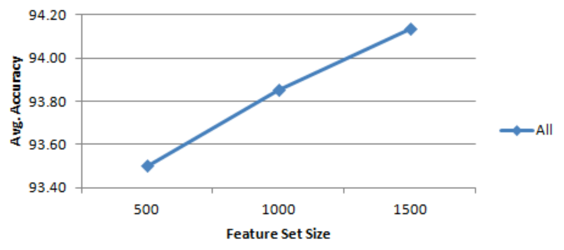

Fig. 5. Total average accuracy of all mutual information classifiers vs. feature set size for 6-grams.

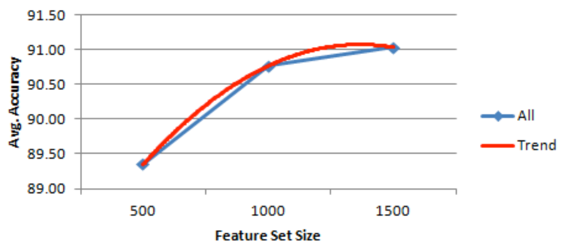

Fig. 6. Total average accuracy of all random projection classifiers vs. feature set size for 6-grams.

B. Feature Set Size Variation

The feature set size parameter was assigned 3 different values, 500, 1000, and 1500, in order to find the impact, if any, of feature set size on the effectiveness of the various classifiers. For this experiment, *n*-gram size was maintained at *6*-grams.

In general, both the mutual information and random projection trained classifiers seemed to make more accurate class predictions as the feature set size increased. The total average classifier accuracy steadily increased as feature set size increased for both the random projection trained classifiers and the mutual information trained classifiers, as depicted in Figure 5 and Figure 6 below.

The total average accuracy of the mutual information trained classifiers appears to grow somewhat linearly with the feature set size parameter. This correlates with the fact that the feature set size was increased linearly in the experiments, by 500 features each gradation. For example, when the feature set size was increased from 1000 to 1500, the mutual information trained SVM classifier experienced an increase in average accuracy that was approximately 89% of the increase it experienced when the feature set size was increased from 500 to 1000. Only a limited number of feature set sizes are examined in these experiments, however, if the feature set size was to be increased linearly, it is expected that the growth rate of the total average accuracy would begin to taper off at some point as features with less predictive information are introduced.

On the other hand, the average accuracy of the random projection trained classifiers appears to have a more polynomial growth rate, where the average accuracy experiences a greater increase when raised from 500 features to 1000 features, than

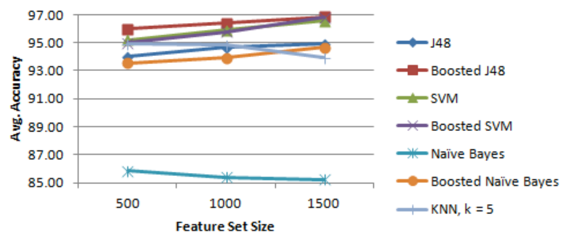

Fig. 7. Average accuracy of each mutual information classifier vs. feature set size for 6-grams.

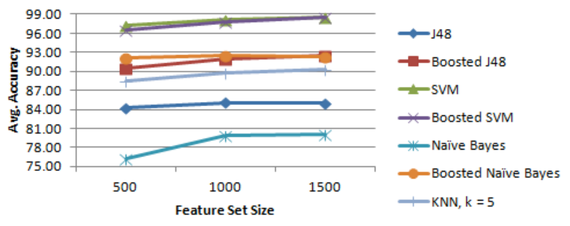

Fig. 8. Average accuracy of each random projection classifier vs. feature set size for 6-grams.

from 1000 features to 1500 features. For example, when the feature set size was increased from 1000 to 1500, the random projection trained SVM classifier experienced an increase in average accuracy that was only about 48% of the increase it experienced when the feature set size was increased from 500 to 1000. This may be attributed to the fact that all of the features in the original data set are represented in the reduced set of features created through random projection; predictive information from each original feature is present in the reduced feature set. That is, increasing the feature set size of the reduced feature set does not introduce the influence of previously absent features from the original feature set, as in the case of mutual information feature reduction, but simply allows for each original feature to have a greater influence i.e. provide more detailed information in the reduced set of features. It follows that since predictive information from all the original features is present, providing a greater degree of information about the same features will most likely help in increasing the accuracy of the classification process. However, it will not introduce new predictive information from other features.

Total average accuracy for mutual information is higher, on average, then the average accuracy of random projection. This is due to the fact that Naïve Bayes classifier performs much worse using random projection, then the mutual information algorithm.

V. Conclusions

Random Projection is a feature reduction tool that can be used to improve speed and accuracy of data mining algorithms. Combinations of *n*-gram analysis, random projection, and data mining algorithms can be used to find malicious executables, as a part of static analysis method.

An interesting trend is observed in the data - almost all the tested classifiers' accuracy tends to cluster while using mutual information, however all of the classifiers perform differently while using random projection.

Overall, the results of the experiments indicate that the best accuracy is achieved using 4-gram feature retrieval method, 1400 feature set generated by random projection algorithm, and Support Vector Machines (SVM) classifier. Boosted SVM classifier performed almost identical, using the same conditions.

VI. Acknowledgements

This material is based upon work supported by the U.S. Air Force, Air Force Research Laboratory under Award No. FA9550-10-1-0289.

References

[1] L. J. Hoffman, *Rogue Programs: Viruses, Worms and Trojan Horses*. New York, NY, USA: Van Nostrand Reinhold Co., 1990.

[2] G. McGraw and G. Morisett, "Attacking malicious code: A report to the infosec research council," vol. 17, no. 5, pp. 33–41, 2000.

[3] J. Durand and T. Atkison, "Applying random projection to the classification of malicious applications using data mining algorithms," *Proceedings of the 50th Annual Southeast Regional Conference*, pp. 286–291, 2012.

[4] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns," *Proceedings of the 12th Conference on USENIX Security Symposium*, p. 12, 2000.

[5] M. Christodorescu, S. Jha, M. D. Preda, and S. Debray, "A semantics-based approach to malware detection," *Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 377–388, 2007.

[6] S. Corporation, "Symantec internet security threat report: Trends for 2012," Apr. 2012. [Online]. Available: http://www.symantec.com/threatreport/

[7] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 470–478, 2004.

[8] ——, "Learning to detect and classify malicious executables in the wild," *The Journal of Machine Learning Research*, vol. 7, pp. 2721–2744, 2006.

[9] D. K. S. Reddy and A. K. Pujari, "N-gram analysis for computer virus detection," *Journal in Computer Virology*, vol. 2, pp. 231–239, 2006.

[10] O. Henchiri and N. Japkowicz, "A feature selection and evaluation scheme for computer virus detection," *Proceedings of the 6th International Conference on Data Mining*, pp. 891–895, 2006.

[11] R. Bellman, *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[12] T. Atkison, "Aiding prediction algorithms in detecting high-dimensional malicious applications using a randomized projection technique," *Proceedings of the 48th Annual ACM Southeast Conference*, 2010.

[13] ——, "Applying randomized projection to aid prediction algorithms in decting high-dimensional rogue applications," *Proceedings of the 47th ACM Southeast Conference*, 2009.

[14] J. Durand and T. Atkison, "Using randomized projection techniques to aid in detecting high-dimensional malcious applications," *Proceedings of the 2011 ACM Southeast Conference*, 2011.

[15] J. Durand, T. Atkison, J. Flores, N. Kraft, and R. Smith, "Using executable slicing to improve rogue software detection algorithms," *International Journal of Secure Software Engineering*, vol. 2, no. 2, pp. 53–64, 2011.

[16] W. Arnold and G. Tesauro, "Automatically generated win32 heuristic virus detection," *Proceedings of the 2000 International Virus Bulletin Conference*, 2000.

[17] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based detection of new malicious code," *Proceedings of the 28th Annual International Computer Software and Applications Conference*, pp. 41–42, 2004.

[18] ——, "Detection of new malicious code using n-grams signatures," *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust*, pp. 193–196, 2004.

[19] J. O. Kephart, G. B. Sorkin, W. C. Arnold, D. M. Chess, G. J. Tesauro, and S. R. White, "Biologically inspired defenses against computer viruses," *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 985–986, 1995.

[20] C. Marceau, "Characterizing the behavior of a program using multiple-length n-grams," *Proceedings of the 2000 Workshop on New Security Paradigms*, 2000.

[21] I. Santos, Y. K. Penya, J. Devesa, and P. Bringas, "N-grams-based file signatures for malware detection," *Proceedings of the 11th International Conference on Enterprise Information Systems*, pp. 317–320, 2009.

[22] R. Perdisci, A. Lanzi, and W. Lee, "Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables," *Proceedings of the Computer Security Applications Conference*, pp. 301–310, 2008.

[23] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Harlow, England: Addison Wesley, 1999.

[24] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," *Proceedings of the 14th International Conference on Machine Learning*, pp. 412–420, 1997.

[25] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pp. 38–49, 2001.

[26] R. Perdisci, A. Lanzi, and W. Lee, "Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables," *Proceedings of the 2008 Annual Computer Security Applications Conference*, 2008.

[27] N. Goel and G. Bebis, "Face recognition experiments with random projection," *Proceedings of SPIE 2005*, 2005.

[28] D. Achlioptas, "Database-friendly random projection," *Proceedings of ACM Symposium on the Principles of Database Systems*, pp. 274–278, 2001.

[29] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, no. 2, pp. 215–245, 1995.

[30] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary Mathematics*, vol. 26, pp. 189–206, 1984.

[31] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel J. Math*, vol. 52, pp. 46–52, 1985.

[32] A. Singhal, "Modern information retrieval: A brief overview," *Bulletin of the Technical Committee on Data Engineering*, vol. 24, no. 4, pp. 35–43, 2001.

[33] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 2005.