# Applying Random Projection to the Classification of Malicious Applications using Data Mining Algorithms

Jan Durand
Department of Computer Science
Louisiana Tech University
Ruston, LA 71270
jrd037@latech.edu

Travis Atkison
Department of Computer Science
Louisiana Tech University
Ruston, LA 71270
atkison@latech.edu

## ABSTRACT
This research is part of a continuing effort to show the viability of using random projection as a feature extraction and reduction technique in the classification of malware to produce more accurate classifiers. In this paper, we use a vector space model with n-gram analysis to produce weighted feature vectors from binary executables, which we then reduce to a smaller feature set using the random projection method proposed by Achlioptas, and the feature selection method of mutual information to produce two separate data sets. We then apply several popular machine learning algorithms including J48 decision tree, naïve Bayes, support vector machines, and an instance-based learner to the data sets to produce classifiers for the detection of malicious executables. We evaluate the performance of the different classifiers and discover that using a data set reduced by random projection can improve the performance of support vector machine and instance-based learner classifiers.

## Categories and Subject Descriptors
D.2.0 [**Software Engineering**]: General – Protection mechanisms

## General Terms
Security

## Keywords
Malicious software detection, information retrieval, n-gram analysis, random projection, data mining

## 1. INTRODUCTION
Since the discovery of the "Elk Cloner" computer virus "in the wild" in 1981[1], the rate of creation and spread of malicious software applications has increased prodigiously. With the advent of the Internet, all types of malicious software or "malware" have an easily accessible and effective medium for propagation to victims' machines. 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.



*ACM SE 12*, March 29 - 31 2012, Tuscaloosa, AL, USA.


Traditionally, these defense tools used signature-based detection which involves comparing the byte sequence of a suspected malicious application to the signatures of known threats, stored in a virus signature database. 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.

There have been several research efforts into 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 [2-5] 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" [2], which refers to the challenges of computing in a high-dimensional space. 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.

This research is part of a continuing effort to show the viability of using random projection as a feature extraction and reduction technique in the classification of malware to produce more accurate classifiers.

## 2. BACKGROUND
Below, we give a brief description of the information retrieval technique of n-gram analysis which we use to produce feature vectors from executable instances, as well as an overview of the dimensionality reduction techniques used to reduce the dimension of those feature vectors.

### 2.1 N-gram Analysis
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 [2-17]. An *n*-gram is any substring of length *n* [18]. Since *n*-grams overlap, they do not just capture statistics about substrings of length *n*, but also implicitly capture frequencies of longer substrings [12]. In our experiments, n-grams are extracted from the hexadecimal byte strings of executable files, and one gram corresponds to one byte in hexadecimal form. Due to the high dimensionality of n-gram feature sets, these research efforts are often subject to the "curse of dimensionality."

[19] Many of these research efforts use some form of dimensionality reduction, such as mutual information in [5], to curb these large feature sets in order to mitigate the effects of the "curse of dimensionality."

Kephart et al. referenced the fields of Information Retrieval and Artificial Intelligence, while working with n-grams and Artificial Neural Networks (ANNs) in their research of the malware detection problem [14, 20]. Abou-Assaleh et al. applied the *Common N-Gram (CNG)* analysis method [21], known for its success in automatic authorship attribution and text clustering, to the malware detection problem [12, 13].

Santos et al. [16] introduced a similar approach to Abou-Assaleh et al. [13], using n-gram signatures to represent malicious and benign programs. The n-gram signature for a program or file consisted of every n-gram present in that file, and signatures for 1000 malicious and 1000 benign binary files were extracted to create the corpus; 66% percent was used for training and 34% for testing. Santos et al. also used the k-Nearest-Neighbor algorithm to classify signatures of new instances against signatures in the corpus.

Kolter used the information retrieval technique of n-gram analysis to create binary feature vectors of malicious and benign programs, which were used with machine learning algorithms to classify unknown program instances [4, 5]. To create the feature vector for a program, n-grams were extracted from the hexadecimal byte sequences of the program, with each byte being considered a gram. Several machine learning algorithms were applied to the feature set, using the Wakaito Environment for Knowledge Analysis (WEKA) [22]. These classification algorithms were the k-Nearest-Neighbors Instance-based learner (IB*k*), a Naïve Bayes classifier, a support vector machine (SVM), a decision tree (J48), and "boosted" versions of the last three classifiers [5]. 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. Table 1 presents the AUC results for the algorithms.

**Table 1. Classifier AUC Results with 95% Confidence Interval from Kolter [5]**

| **Method** | **AUC** |
|---|---|
| Boosted J48 | 0.9958±0.0024 |
| SVM | 0.9925±0.0033 |
| Boosted SVM | 0.9903±0.0038 |
| IB*k*, *k* = 5 | 0.9899±0.0038 |
| Boosted Naïve Bayes | 0.9887±0.0042 |
| J48 | 0.9712±0.0067 |
| Naïve Bayes | 0.9366±0.0099 |

## 2.2 Mutual Information

Mutual information has been used as a basis for feature selection in the field of machine learning. Due to the enormity of the set of n-grams extracted from a program, Kolter introduced a dimensionality reduction processing step using mutual information 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. [23], also known as the Information Gain, calculated as:

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

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 $C_i$ [5].

## 2.3 Random Projection

Malicious application detection, following the genre of information retrieval, suffers from the problem that the data, once processed, is encoded in extremely high dimensions. This high-dimensional data limits the kind and amount of analysis that can be preformed. Though the feature selection technique of mutual information has been very popular in reducing the feature sets of such 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 [5], which selects a subset of the total feature set, 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" [24], thus creating a completely new set of features. This type of projection attempts to retain the maximum amount of information embedded in the original feature set while substantially reducing the number of features required. By reducing the number of features, greater amounts of analysis can be performed. The core concept has been developed out of the Johnson-Lindenstrauss lemma [25] which states that any set of n points in a Euclidean space can be mapped to $\mathbb{R}^t$ where $t = \frac{O(\log n)}{\varepsilon^2}$ with distortion $\leq 1 + \varepsilon$ in the distances. Such a mapping may be found in random polynomial time. A proof of this lemma can be found in [26].

Random projection has been used in several other research efforts. Mannila et al. used random projection to aid in finding similarities between sequences of events, specifically looking at how to better handle network alarms within the telecommunication field [27] . Bingham et al. utilized random projection in image and text information retrieval and showed that random projection offered greater accuracy and computational savings than more traditional dimensionality reduction techniques such as principal component analysis and singular value decomposition [28].

Kaski [29] utilized random projection in a text retrieval application using WEBSOM, a graphical self-organizing map, to overcome the computational costs of traditional dimensionality reduction methods like principal component analysis on high-dimensional data sets. Kaski was able to improve the classification and topic separation performance of his tool after using random projection over the previous methods used.

A few research efforts have used random projection with the dimensionality reduction method of latent semantic indexing [30-32], including Papadimitriou et al. who successfully used random projection as a preprocessing step to the latent semantic indexing method and improved the asymptotic running time of their overall system from O(*mnc*) to O($m(log^2n + clogn)$), where *m* and *n* are the dimensions of the matrix and *c* is the average number of terms per document [30].

Goel et al. used random projection in the field of facial recognition, which yielded comparable results to that of principal component analysis while being computationally less expensive and data independent [24].

Li et al. applied random projections to the field of network anomaly detection to more precisely identify the underlying causes of network anomalies [33]. More specifically, their technique, called "Defeat", allowed for the identification of the

IP-flow(s) responsible for the anomaly as opposed to the traditional method which only identified the origin-destination flow. This was accomplished by creating multiple random projections or "sketches" of global network traffic in the form of IP-flows, which were then mined for information using the "subspace" method. A voting procedure was also applied to the detection results from the sketches to increase detection rate while reducing false alarms. Based on an evaluation of a week-long trace, Defeat displayed detection rates comparable to previous methods and detected nearly 200 more anomalies. Li et al. concluded "that random projections appear to preserve properties of traffic data that are important for the effectiveness of the subspace method." [33]

Atkison was the first to introduce the random projection feature extraction technique to the realm of malicious application detection [7-11]. In a recent paper [10], Atkison et al. used a vector space model with n-gram analysis to produce weighted feature vectors from binary executables, similarly to Kolter [5]. 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 with a Gaussian distribution of mean 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 [34]; 3) and random set projection based on the Linial-London-Rabinovich algorithm [35], which is an extension of the Johnson-Lindenstrauss [36] and Bourgain [36] algorithms.

To test the efficacy of using random projection in this particular context of malware detection, Atkison extracted n-gram feature vectors, with n-grams of length 3, 5, and 7, 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. For each of these corpuses, Atkison compared each document feature vector to every other feature vector and classified the document vector 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." [38] 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 [10], comparable to results of previous research efforts using the reduction technique of mutual information.

# 3. EXPERIMENT

In our experiments, we follow the methodology used by Kolter [5] and use some well-known data mining algorithms to produce classifiers to detect whether an executable is malicious or benign. However, we use random projection as the dimensionality reduction technique in order to compare the performance of random projection against the established mutual information method. The following provides a description of the components of the experimental methodology that was used to detect malicious applications using the information retrieval technique of *n*-gram analysis and the dimensionality reduction technique of random projection, as well as descriptions of the various data mining algorithms used.

## 3.1 Data Set

The data set that was compiled together 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.

A Trojan horse, similar to the myth, may provide a useful service (for example, a calculator or Notepad) but once executed performs harmful actions. Symantec reported in their bi-annual threat report for the first half of 2005, that "six of the top ten spyware (information leakage) programs were delivered to their victim by being bundled with some other program." [30] Trojans are a very popular and effective way of infiltrating user systems. To give an idea of their prevalence, in 2009, Trojans accounted for 6 of the top 10 new malicious code families detected; 51 percent of the volume of the top 50 malicious code samples reported; four of the top 10 staged downloaders; and eight of the top 10 threat components downloaded by modular malicious software [39].

## 3.2 Data Mining Algorithms

The following classification algorithms were used in the performance evaluation of random projection as an effective dimensionality reduction technique in the detection of malicious applications. All classifiers used in this research were trained and tested using the Waikato Environment for Knowledge Analysis (WEKA) machine learning software suite [22], with 10-fold cross validation on the training data set.

### *3.2.1 Instance Based Classifier*

The instance-based classifier, also known as a lazy learner, uses one of the most simple classification algorithms. It learns by storing the vectors of training examples and their corresponding labels, and classifies an unknown instance by giving it the label of the most similar instance in the training set. The Euclidean distance is usually used to measure the similarity or dissimilarity of two instances. Instance-based learners are also known as 'nearest neighbor' or 'k-nearest-neighbors'. The k-nearest-neighbor algorithm classifies an instance by a majority vote of the labels of the k most similar instances in the training set. In our experiments we used an instance-based learner (IBk) with a k of 5 as was used in Kolter's experiments, using Euclidean distance as the distance metric.

### *3.2.2 Naive Bayes Classifier*

The naive Bayes classifier is a probabilistic classifier based on the Bayesian theorem and is popular in the fields of Information

retrieval and Machine learning. This classifier uses the prior probability of a class $P(C_i)$ and the conditional probability of each feature attribute for the specified class, $P(f_j|C_i)$, in order to determine the posterior probability that an unknown instance belongs to a particular class given its feature vector, $P(C_i, f)$ 5]. The prior probabilities of the classes and the conditional probabilities of feature attributes for each class are estimated by counting the number of occurrences in the training data of each class and the attribute values for each class. The classifier determines the class of an unknown instance by selecting the class with the highest posterior probability. In order to simplify the modeling of $P(f_j, C_i)$, it is assumed that the conditional probabilities of each attribute are independent [40].

$$P(C_i|f) = P(C_i)\prod_j P(f_j|C_i) \quad \textbf{(2)}$$

### *3.2.3 Decision Tree*

Decision trees or classification trees consist of internal nodes which represent instance attributes, and leaf nodes which represent the possible classes or labels of an instance. In order to classify an unknown instance, the classifier starts at the root node and follows the branch to a node which has an attribute value or range of values that corresponds to the attribute value of the instance in question. The tree is traversed in this fashion until a leaf node is reached, and then the instance is classified as the label of that leaf node. A decision tree is created by considering all the instances in the training set and selecting a feature attribute which best splits the data set into its respective classes. The attribute that was previously selected is then removed from future consideration and the process is repeated recursively on each data subset. The last attribute split creates the leaf nodes which are labeled based on a majority vote of the class labels of its elements. For attribute selection, most implementations of the decision tree use an information gain-based measure such as gain ratio in the C4.5 algorithm developed by Quinlan [41]. J48 is a Java implementation of the C4.5 algorithm available in WEKA [22].

### *3.2.4 Support Vector Machines*

Support vector machines (SVMs) produce a binary linear classifier which is able to separate a data set into two distinct classes, the positive and the negative class. The method works by using a kernel function to map the feature vectors of the data instances into a higher dimension so that the two classes of instances can be easily separated by a straight line or hyper plane. The hyper plane is described by the equation $(\vec{w}.\vec{x}) - b = 0$, where $\vec{w}$ is a vector or set of weights perpendicular to the hyper plane; $\vec{x}$ is a point belonging to the hyper plane; and be is a threshold value which determines the offset of the hyper plane from the origin, along $\vec{w}$. The algorithm strives to choose a $\vec{w}$ and $b$ which create an optimal hyper plane that maximizes the margin between itself and the closest instances of each class. The idea is to increase the dimension of the training data to get the instances of one class on one side of the hyper plane and the next class on the other side. Classification is performed by mapping an unknown instance into the higher feature dimension and labeling it based on which side of the hyper plane it appears; the instance is labeled as the positive class if $(\vec{w}.\vec{x}) - b > 0$, or as the negative class if otherwise. WEKA implements the sequential minimal optimization (SMO) algorithm for training SVMs.

### *3.2.5 Boosted Classifiers*

Boosting is a way of combining several classifiers to produce a better classifier. Research shows that ensemble classifiers usually perform better than individual classifiers, and are able to improve regardless of how weak the individual classifier is [42]. Just like in Kolter [5], we used the AdaBoost (Adaptive Boosting) method implemented in WEKA to boost the performance of each of the aforementioned classifiers except the instance-based learner due to computational expense. AdaBoost works by iteratively training several models of a classifier and reweighting the data set instances in favor of instances which were misclassified previously so subsequent classifiers can better predict the classes of those instances. During classification of an unknown instance, the different models and their respective weights are used to select the class with the highest weight.

## 3.3 Experimental Design

This section describes the overall design of this experiment. Following the methodology of Kolter [5], an n-gram size of 4 was used to create feature vectors from the executables in the data set. A binary value-weighting scheme was used for this effort, whereby 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. Feature vectors with a dimension of 25,368,317 were created from the data set using n-grams of length 4. 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.

For the dimensionality reduction portion, two sets of reduced corpuses were created. One set of reduced vectors was created by reducing the dimensionality of each vector to 500 features via mutual information. The other set of feature vectors was reduced to 200,000 features via mutual information before being further reduced to 500 features using the method of random projection proposed by Achlioptas [34]. The mutual information preprocessing phase was used to remove the influence of insignificant features and also speed up the overall reduction process.

Both of these data sets were then used to train and test classifiers created using the naïve Bayes, Instance-based learner, Support Vector machines, and J48 decision tree classification algorithms, as well as boosted versions of the naïve Bayes, support vector machine, and J48 algorithms.

Each of the classifiers was trained and tested using 10-fold cross validation. That is, the data set was separated into 10 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. The results obtained from these experiments are presented below.

# 4. RESULTS

Both data sets created classifiers which produced promising results. In tables 2 and 3 below, the true positive (TP) rate represents the number of malicious instances classified as malicious; the false positive (FP) rate represents the number of benign instances classified as malicious; the area under the curve (AUC) measure represents the area under the receiver operating characteristic (ROC) curve generated by the classifier; the accuracy represents the number of instances correctly classified by the classifier; and the classifiers are listed in descending order of accuracy.

The true positive rate gives us an idea of how effective a classifier is at detecting malicious executables, however, we need to take

into account the false positive rate to ensure that our classifier does not flag too many benign executables as malware. That is, in a real computing environment there will be many more benign executables than malware, therefore a detection solution which generates too many false alarms can severely disrupt a computer user's experience, potentially making the solution more unattractive than the problem.

**Table 2. Performance values of classifiers produced with mutual information-reduced data set**

| Method | TP Rate | FP Rate | AUC | Accuracy % |
|---|---|---|---|---|
| Boosted J48 | 0.967 | 0.034 | 0.994 | 96.67 |
| SVM | 0.966 | 0.037 | 0.965 | 96.49 |
| Boosted SVM | 0.966 | 0.037 | 0.965 | 96.49 |
| Boosted N. Bayes | 0.954 | 0.041 | 0.986 | 95.63 |
| IBk, k = 5 | 0.977 | 0.075 | 0.99 | 95.50 |
| J48 | 0.954 | 0.053 | 0.939 | 95.13 |
| Naïve Bayes | 0.796 | 0.113 | 0.898 | 83.42 |

Considering the classifiers that were trained and tested with the data set reduced by mutual information, the performance results were similar to those obtained by Kolter [5]. Boosted J48 had the highest accuracy and area under the ROC curve (AUC) at 96.67% and 0.994 respectively, and lowest false positive rate at 0.034. The SVM and boosted SVM classifiers also performed very well with results close to that of the boosted J48 classifier. The other classifiers all performed comparably except for the naïve Bayes classifier which had the worst scores of the mutual information set of classifiers, just as in Kolter's results, with a true positive rate of 0.796, a false positive rate of 0.113, an AUC of 0.898, and an accuracy of 83.42%. It should be noted that though the instanced-based learner had the highest true positive rate and AUC, it also had the second highest false positive rate, thus reducing its overall accuracy.

**Table 3. Performance values of classifiers produced with random projection-reduced data set**

| Classifier | TP Rate | FP Rate | AUC | Accuracy % |
|---|---|---|---|---|
| SVM | 0.986 | 0.025 | 0.981 | 98.15 |
| Boosted SVM | 0.972 | 0.034 | 0.986 | 96.98 |
| IBk, k = 5 | 0.997 | 0.097 | 0.993 | 95.75 |
| Boosted N. Bayes | 0.945 | 0.053 | 0.973 | 94.57 |
| Boosted J48 | 0.927 | 0.074 | 0.979 | 92.66 |
| J48 | 0.892 | 0.166 | 0.841 | 86.75 |
| Naïve Bayes | 0.85 | 0.282 | 0.829 | 79.47 |

In contrast to the results from the mutual information group of classifiers, the classifiers generated from the data set reduced via random projection produced results which promoted some classifiers while demoting others. In this group of classifiers, the SVM classifier performed the best with a TP rate of 0.986, an FP rate of 0.025, an AUC of 0.981, and an overall accuracy of 98.15%, nearly 2% higher than the best performer in the mutual information group, the boosted J48 classifier. The boosted SVM and the IBk classifier both had improved results, except for the increased FP rate of the IBk classifier, following in 2nd and 3rd place respectively. The boosted J48 classifier performed worse on the random projected data set, only besting the J48 and naïve Bayes classifiers; the naïve Bayes classifier retained its last place position, performing worse than its mutual information-trained counterpart, with a reduced overall accuracy of 79.47%.

# 5. CONCLUSIONS

The results obtained in the experiments above demonstrate that the dimensionality reduction technique of random projection can be used to improve the performance of some data mining classification algorithms. Though the performance of the J48 classifier, naïve Bayes classifier, and their boosted counterparts was reduced when using data instances reduced via random projection as opposed to mutual information, the SVM, boosted SVM, and IBk classifiers experienced an increase in performance. This resulted in the SVM classifier exhibiting the best performance out of all the classifiers across both data sets. This may be due to the fact that SVM and k-nearest-neighbors algorithms utilize a vector space model in the classification process. That is, they are capable of taking advantage of the pair-wise distance preservation characteristic of the random projection technique.

In addition, while the mutual information reduced data set only selected the top 500 n-grams based on information gain as its feature set, the random projected data set created 500 new features incorporating the attributes of a set of 200,000 n-grams. The added information of the other 199,500 n-grams may have also contributed to the increased performance of the SVM and IBk classifiers. The boosted SVM classifier did not outperform the SVM classifier but Kolter suggests that this may be due to the stability of SVMs in classification tasks [6]. Bauer and Kohavi also suggest that boosting can adversely affect stable classifiers [42].

# 6. 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.

# 7. REFERENCES

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

[2] Arnold, W. and Tesauro, G. Automatically Generated Win32 Heuristic Virus Detection. In *Proceedings of the 2000 International Virus Bulletin Conference* ( Sep. 2000).

[3] SYMANTEC CORPORATION. *Symantec Internet Security Threat Report: Trends for 2010*. Symantec Corporation, 2010.

[4] Kolter, J. Z. and Maloof, M. A. Learning to Detect Malicious Executables in the Wild. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Seattle, WA Aug. 2004), ACM Press, 470-478.

[5] Kolter, J. Z. and Maloof, M. A. Learning to Detect and Classify Malicious Executables in the Wild. *The Journal of Machine Learning Research*, 7 (2006), 2721-2744.

[6] Reddy, D. K. S. and Pujari, A. K. N-gram Analysis for Computer Virus Detection. *Journal in Computer Virology*, 2, 3 (2006), 231-239.

[7] Atkison, T. Applying Randomized Projection to Aid Prediction Algorithms in Decting High-Dimensional Rogue Applications. In *Proceedings of the 47th ACM Southeast Conference* (Clemson, SC, USA 2009).

[8] Atkison, T. *Using Random Projections for Dimensionality Reduction in Identifying Rogue Applications*. Mississippi State University, Starkville, MS, USA, Aug. 2009.

[9] Atkison, T. Aiding Prediction Algorithms in Detecting High-Dimensional Malicious Applications Using a Randomized Projection Technique. In *Proceedings of the 48th Annual ACM Southeast Conference* (Oxford, MS, USA Apr. 2010).

[10] Atkison, T. and Durand, J. Using Randomized Projection

Techniques to Aid in Detecting High-Dimensional Malcious Applications. In *Proceedings of the 2011 ACM Southeast Conference* (Kennesaw, GA, USA 2011).

[11] Atkison, T., Durand, J., Flores, J., Kraft, N., and Smith, R. Using Executable Slicing to Improve Rogue Software Detection Algorithms. *International Journal of Secure Software Engineering*, 2, 2 (2011), 53-64.

[12] Abou-Assaleh, T., Cercone, N., Keselj, V., and Sweidan, R. N-gram-based Detection of New Malicious Code. In *Proceedings of the 28th Annual International Computer Software and Applications Conference* ( Sep. 2004), 41-42.

[13] Abou-Assaleh, T., Cercone, N., Keselj, V., and Sweidan, R. Detection of New Malicious Code using N-grams Signatures. In *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust* (New Brunswick, Canada 2004), 193-196.

[14] Kephart, J. O., Sorkin, G. B., Arnold, W. C., Chess, D. M., Tesauro, G. J., and White, S. R. Biologically Inspired Defenses Against Computer Viruses. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (San Francisco, CA 1995), 985-986.

[15] Marceau, C. Characterizing the Behavior of a Program using Multiple-Length N-grams. In *Proceedings of the 2000 Workshop on New Security Paradigms* (Ballycotton, County Cork, Ireland 2000).

[16] Santos, I., Penya, Y. K., Devesa, J., and Bringas, P. N-grams-based File Signatures for Malware Detection. In *Proceedings of the 11th International Conference on Enterprise Information Systems* ( 2009), 317-320.

[17] Perdisci, R., Lanzi, A., and Lee, Wenke. McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables. In *Proceedings of the Computer Security Applications Conference* ( Dec. 2008), 301-310.

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

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

[20] Tesauro, G., Kephart, J. O., and Sorkin, G. B. Neural Networks for Computer Virus Recognition. *IEEE Expert*, 11, 4 (Aug. 1996), 5-6.

[21] Keselj, V., Peng, F., Cercone, N., and Thomas, C. N-gram-based Author Profiles for Authorship Attribution. *Computational Linguistics*, 3 (2003), 255-264.

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

[23] Yang, Y. and Pedersen, J. O. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the 14th International Conference on Machine Learning* ( 1997), 412-420.

[24] Goel, N. and Bebis, G. Face Recognition Experiments with Random Projection. In *Proceedings of SPIE 2005* (Orlando, FL, USA Mar. 2005).

[25] Sallis, P. and MacDonell, S. Software Forensics: Old Methods for a New Science. In *Proceedings of Software Engineering: Education and Practice* ( 1996), 367-371.

[26] Gray, A. R., Sallis, P., and MacDonell, S. *Software Forensics: Extending Authorship Analysis Techniques to Computer Programs*. University of Otago, New Zealand, Dec. 1997.

[27] Mannila, H. and Seppnen, J. K. Finding Similar Situations in Sequences of Events. In *Proceedings of the First SIAM International Conference on Data Mining* ( 2001).

[28] Bingham, E. and Mannila, H. Random Projection in Dimensionality Reduction: Applications to Image and Text Data. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* ( 2001), 245-250.

[29] Kaski, S. Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering. In *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks* (Anchorage, AK, USA May 1998), 413-418.

[30] Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. Latent Semantic Indexing: A Probabilistic Analysis. *Journal of Computer and System Sciences*, 61, 2 (2000), 217-235.

[31] Kurimo, M. Indexing Audio Documents by using Latent Semantic Analysis and SOM. In Oja, E. and Kaski, S., eds., *Kohonen Maps*. Elsevier, 1999.

[32] Lin, J. and Gunopulos, D. Dimensionality Reduction by Random Projection and Latent Semantic Indexing. In *Proceedings of the Text Mining Workshop at the 3rd SLAM International Conference on Data Mining* ( 2003).

[33] Li, X., Bian, F., Crovella, M., Diot, C., Govindan, R., Iannaccone, G., and Lakhina, A. Detection and Identification of Network Anomalies using Sketch Subspaces. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (New York, NY, USA 2006).

[34] Achlioptas, D. Database-friendly Random Projection. In *Proceedings of ACM Symposium on the Principles of Database Systems* ( 2001), 274-278.

[35] Linial, N., London, E., and Rabinovich, Y. The Geometry of Graphs and some of its Algorithmic Applications. *Combinatorica*, 15, 2 (1995), 215-245.

[36] Johnson, W. B. and Lindenstrauss, J. Extensions of Lipschitz Mappings into a Hilbert Space. *Contemporary Mathematics*, 26 (1984), 189-206.

[37] Bourgain, J. On Lipschitz Embedding of finite Metric Spaces in Hilbert Space. *Israel J. Math*, 52 (1985), 46-52.

[38] Singhal, A. Modern Information Retrieval: A Brief Overview. *Bulletin of the Technical Committee on Data Engineering*, 24, 4 (2001), 35-43.

[39] Chen, X., Francia, B., Li, M., McKinnon, B., and Seker, A. Shared information and Program Plagiarism Detection. *IEEE Transactions on Information Theory*, 50, 7 (2004), 1545-1551.

[40] Lewis, David. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In Nedellec, Claire and Rouveirol, Celine, eds., *Machine Learning: ECML-98*. Springer Berlin/Heidelberg, 1998.

[41] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[42] Bauer, E. and Kohavi, R. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36, 1 (Jul. 1999), 105-139.