2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems

Detecting Worms Using Data Mining Techniques : Learning in the Presence of Class Noise

Ismahani Ismail Muhammad Nadzir Marsono Sulaiman Mohd Nor

Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 Johor Bahru, Johor, Malaysia
ismahani@fke.utm.my, nadzir@fke.utm.my, sulaiman@fke.utm.my

***Abstract*—Worms are self-contained programs that spread over the Internet. Worms cause problems such as lost of information, information theft and denial-of-service attacks. The first part of the paper evaluates the detection of worms based on content classification by using all machine learning techniques available in WEKA data mining tools. Four most accurate and quite fast classifiers are identified for further analysis–Naive Bayes, J48, SMO and Winnow. Results show that classification using machine learning techniques could classify worms to 99% accuracy. From the accuracy perspective, J48 performs better than other algorithms meanwhile Naive Bayes and Winnow show the best performances in terms of speed. The second part of the paper analyzes the accuracy these four classifiers under the presence of class noise in learning corpora. By injecting class noise ranging between 0% and 50% into positive and negative corpora, results from the simulation show gradual decrease in accuracy and increase in false positive and false negative for all analyzed techniques. The presence of the classes noise affects false positive more significantly compared to false negative. The results show that worm detection with classification algorithms could not tolerate the presence of classes noise in learning corpora.**

***Index Terms*—data-mining techniques, worm detection, class noise**

I. INTRODUCTION

Worms are self-contained programs that spread over the Internet, usually by exploiting vulnerabilities in heterogeneous software running on networked computers [1]. Worms cause loss of information, discontinuity access, unauthorized change of data, systems, networks or services [2], information theft [3] and denial-of-service attacks [4]. For example, Sapphire worm creates so much network traffic in a short time that it overwhelms routers and other network nodes [4].

Each worm has a different way to spread. For instance, internet worms spread through copying themselves to networked resources, exploiting operating system vulnerabilities, and penetrating public networks [5]. On the other hand, email worms spread via infected email messages in the form of an attachment or links to infected websites. These worms harvest email addresses from victim machines in order to spread further. Instant Messaging worms spread using instant messaging applications by sending links to infected websites to everyone on the local contact list [6]. Peer–to–peer worms copy themselves into a shared folder by placing a copy of itself under a harmless name.

An important characteristic of these worms is their active propagation through networks. Although vulnerability being exploited may vary between worms, this characteristic exists in all worm types and should be considered when attempting to detect any such activity. Because of this characteristic, this paper attempts to show that content prevalent of worms could be detected by data–mining techniques on worms content by using machine learning techniques. We also investigate the detection performance in terms of pruning features in the dataset to improve the detection performance. From our investigation, four most accurate and quite fast classifiers are Naive Bayes, J48, SMO and Winnow. These classifiers could classify worms with 99% accuracy.

Generating large and timely training set is a complex task, especially due to the needs for recent and the comprehensive corpora. The samples of worm sometimes do not reflect current attacks, often do not include of all types of worms. Another possible approach is using classified files (either worm or normal) as training data set. However, since no worm detectors ever achieve 100% accuracy, noisy learning corpora (termed class noise, where worm trace files are present in normal corpus and likewise, normal trace files in worm corpus) may affect worm detection accuracy. This paper analyzes the effects of contaminated learning corpora on the accuracy of worm detection using Naive Bayes, J48, SMO, and Winnow classifiers. Our simulation results in WEKA [7] show gradual decrease in detection accuracy with increases in classes noise. This shows that worm detection using these four classifiers could not tolerate the presence of classes noise in the learning corpora.

The rest of the paper is organized as follows. Section II discusses related works in detecting worm using data mining techniques. This section also discusses the classification with class noises. Section III discusses the structure of worms. Section IV shows the work flows for supervised learning worm classification. The experiments are elaborated in Section V. Section VI discusses the learning models for classification under presence of class noise and their analyse in Section VII. Conclusion and future work are in Section VIII.

978-0-7695-4319-2/10 $26.00 © 2010 IEEE
DOI 10.1109/SITIS.2010.41

II. Related Works

There have been few attempts to use data mining techniques for the purpose of identifying new or unknown malicious code including worm. These techniques learn the distinctions among different positive and negative classes. Once trained with examples to form a generative model, a supervised-learning technique could recognize the exact or similar patterns observed during learning. Payload features could be extracted from the content of a trace file, which could be characters, byte strings, words, phrases or n-grams. Worm detection using content classification techniques have shown to achieve 99% accuracy, but none has achieved 100% detection accuracy [8]–[10]. The learning corpora are usually labeled manually by human (experts) and are considered correctly labeled (i.e., without classes noise in learning corpora).

Machine learning

The work proposed in [8] proposes a data mining framework to detect new, previously unseen malicious executables by finding patterns in observed dataset. These patterns are used to detect a set of new malicious binaries. In [8], the authors compare the detection accuracy between signature-based method and three data mining algorithms: RIPPER, naive Bayes and multi-classifier to detect unknown malicious executables. Their results showed that detection accuracy of the data mining algorithms could achieve more than 97%, which is over two times the detection accuracy of signature-based methods.

A similar work in [9], applies several learning methods: Instance-Based k (IBk), Term Frequency-Inverted Document Frequency (TFIDF), naive Bayes, Support Vector Machine (SVM), and J48. They also applied boosted classifier for naive Bayes, SVM, and J48 classifiers. Their experimental results suggest that methods of text classification are appropriate for detecting malicious executables. Their experiment results show that the boosted J48 give the best worm detection accuracy. They suggest that boosting improves the performance of unstable classifiers, such as J48, by reducing their bias and variance.

Work in [11] proposes a worm attack model called OS-JUMP to detect polymorphic worm. Their method detected the worm through recognized JUMP address using signature-based method and data mining algoritms; Bayes and Artificial Neural Network (ANN). Their work shows that signature-based method failed to detect polymorphic worm because the JUMP address changed but Bayes and ANN can still detect this type of worms. Work in [12] presented a new approach based on ANN for detecting the presence of worm based on the computers behavioral measures. They suggest that the ANN approach has computational advantages when real-time computation is needed, and has the potential to detect previously unknown worm.

Research in [1] focuses in classifying structural and behavioural data by using data mining algorithms. Using WEKA, their experiment results show that decision tree classifier offers better classification than naive Bayes classifier for all structural data. Moreover, their experiments show that the classification accuracy for behavioural data is much less than the structural data. Work in [10] uses variable length of instruction sequences that extracted from the disassembly of worms and clean programs as the features to be classified by data-mining techniques- Decision tree, Random Forest and Bagging. They show their simulation result could achieve to 96% of detection accuracy.

Machine learning with presence of class noise

There are two types of classification noise - class noise [13] and attribute noise [14]. Class noise is mislabeled examples, where worm trace files are labeled as normal (i.e., false negative) and likewise, normal trace files as worm (i.e., false positive). The likelihood of the presence of mislabeled examples in learning corpora is high. There are two ways to cushioning the impacts of class noise. First is to remove class noise by filtering the mislabeled examples [13] and second is to develop classifiers that are tolerant to class noise [13], [15], [16]. In the latter approach, multi-classifier ensemble is generally used to avoid class noise influence on the decision [17]. A variation of naïve Bayes classifier has been proposed in [13] that models the learning under the presence of attribute noise, class noise, or both.

Based on these researches, machine learning algorithms are effective enough to be applied as worm detection method. Our work extends the use of supervised-learning techniques under the present of class noise. In this paper, we attempt to investigate the availability of all machine learning techniques in WEKA for worm content–based detection. Our work considers the time of processing and accuracy factors of the classification techniques. The classifiers that could detect the worm content faster and give higher accuracy are listed for more in-depth analysis. We observe the detection accuracy, false positive and false negative for each classification techniques investigated.

III. Worm Structure and Characteristics

Network-oriented infection strategy is the primary characteristic of computer worms. The generic structures of computer worms include essential components such as target locator and infection propagator modules, and other nonessential modules such as the remote control, update interface, life-cycle manager, payload routines, and self–tracking [18] (See Fig. 1).

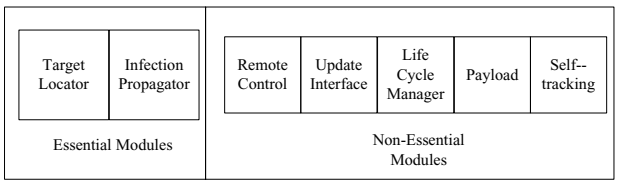

Fig. 1. Worm Generic Structure [18]

Worm must have a target locator segment in order to spread. Most email worms search for e-mail addresses and simply send copies of themselves to such addresses. This could be an easy way for worm to spread further. On the other hand, most of IP

scanning worm variants have scanning mechanisms to scan the network for nodes on the IP level and to check for vulnerable systems.

Another very important component of the worm is an infection propagator, which is used to transfer itself to a new node. Typically, the attacker tricks the recipient into executing the worm based on social engineering techniques [19]. Some worms are equipped to remotely control infected systems. This remote control module allows the authors to issue commands to the worm such as for triggering DDoS (distributed denial of service) tool against several unknown targets [4]. Without such a module, the worms author cannot control the worm ever when sending control messages to the worm copies. However, more and more worms deploy several exploit modules to execute the worm automatically on the vulnerable remote system without users intervention.

Some worms variants include a life-cycle manager to control the ability of the worm to remain active in network [18]. Many worms have bugs in their life-cycle manager which results in worms that run without ever stopping. Some advanced worms have an update or plug-in interface feature to update the worms code on already-compromised systems. The attacker is interested in changing any detectable structure and behavior of the worm and even sending new infection strategies to as many compromised nodes as possible [18].

Another optional, but common component of a computer worm is its payload. In many cases, computer worms do not contain any payload. Examples of such payloads include data destruction, messages with insulting text or spurious e-mail messages sent to a large number computer victims [20]. Some worm authors are interested in seeing how many machines have been infected. To do this, they include in worm self-tracking modules. Computer worms typically send the authors an e-mail message with information about the infected computer such as computer name, IP address, user information and user e-mail address [19] to allow the author track the worms' spread.

These essential worm modules, although sometimes encrypted or mutating, could show prevalent content that could be detected by looking at fixed substrings. By utilizing these prevalence behaviour, worm may be detected based on this information. Since fixed signatures may vary from a worm variant to the other, the use of n–gram has been used in few proposals [9], [21].

IV. Supervised-Learning Worm Classification

Several works have proposed detetion of worm using machine learning [8]–[10]. Fig. 2 shows worm detection using data mining techniques. These techniques require steps as described below.

Feature Extraction

Packet trace files can be extracted to different features such as characters, words, phrases, byte–strings and n-grams.

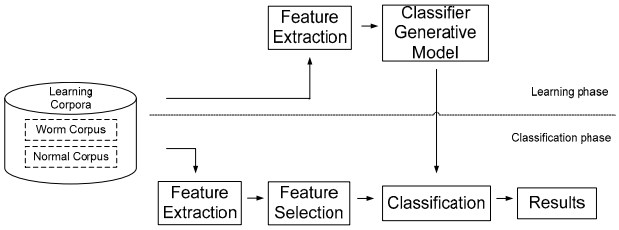

Fig. 2. Worm classification by using data mining techniques

In this research, we extract the executables to n-gram features by combining each n–bytes slice into a longer string. For instance, a byte sequence is `40 65 63 68 6F 20`, the corresponding n-grams are `40656368`, `6563686F` and `63686F20`, respectively, for n=4. Features can be selected (pruned) to reduce the vocabulary size to improve the classification accuracy and time. The feature selection uses certain selection criteria to choose the most informative features. In this work, we use information gain technique from [7] to rank the attributes (features). Information gain (IG) is measured as the amount of the entropy (H) difference when an attribute contributes the additional information about the class. The following is the information gain of attribute Xi for the class C:

$IGi = H(C) - H(C|Xi)$

$H(C) = -\sum p(c) log p(c)$, $c \in C$

$H(C|Xi) = -\sum p(x) \sum p(c|x) log p(c|x)$, $x \in Xi, c \in C$

In training set, we compute the IG for each unique attribute and then remove the attributes whose IG less than predetermined threshold.

Generative Model Formation

Generative model is a model that is generated based on observable data. In machine learning, the classifiers model are built based on observation of labeled features. Using a model that was trained with pre-defined labeled features, a set of features could be estimated to belong to the worms class or otherwise.

Classification

Classification process is a procedure in which individual features are placed into groups based on quantitative information on one or more characteristics inherent in the features. It is based on a training set of previously labeled features (generative model). In this work, we use all machine learning techniques that are provided in WEKA [7]. It contains a collection of machine learning algorithms for data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection.

V. Experimenting with Worm Trace Files

This section describes the experimental work to evaluate the accuracy of worm detection through content classification

using data mining techniques that available in WEKA (version 3.6) [7].

A. *Simulation setup*

Experiments conducted for this work are applied for collection of trace files that contain the payload. The trace files used for the experiments consist of 1194 normal and 1746 worm (malicious) packet trace files. The sources of these trace files are from [22]–[24]. Using Wireshark [25], we convert the *pcap* trace files format to the text format. We extract the packet byte part only. Then, we separate each packet to individual file. The byte sequences are extracted to n-gram features for n = 4. References [9], [21] show that n = 4 produced the best result for the classification performance.

All of the files are separated to two folders named according to their classes. These folders are then uploaded to WEKA in the forms of <relation, attribute, data> where we applied filtering algorithms to convert the data format for training and testing WEKA's classification algorithms. Using *StringToWordVector* option, the number of attributes is set to 1000. We use the nominal attributes since not all classification algorithms can perform well for different types of attributes.

These <relation, attribute, data> sets are classified with all available classifiers in WEKA. The classifiers that could give the best performance in terms of accuracy and processing time are chosen as shortlisted candidates.

For future selection, we use information gain method as the selection algorithm. We parameterized $0.0001 \leq IG \leq 0.5$. For each threshold value, we remove the attributes whose IG is less than the IG value and produce a new file with pruned attributes. Then, we re-apply the chosen classifiers for all pruned data set and examined the accuracy.

B. *Evaluation Criterion*

For the purpose of evaluating the results, we use confusion matrices that were created for each classifier. The following four define the members of the matrix:
$n_{w\to w}$: Number of correctly identified worm payload.
$n_{l\to w}$: Number of wrongly identified normal payload.
$n_{l\to l}$: Number of correctly identified normal payload.
$n_{w\to l}$: Number of wrongly identified worm payload.

- **True Positive** is defined as the ratio of worm payload correctly classified as worm payload. It is given by

$$TP = \frac{n_{w\to w}}{n_{w\to w} + n_{w\to l}} \tag{1}$$

- **False Positive** is defined as the ratio of normal payload that were incorrectly classified as worm payload. It is given by

$$FP = \frac{n_{l\to w}}{n_{l\to w} + n_{l\to l}} \tag{2}$$

- **True Negative** is defined as the ratio of normal payload that were correctly classified as normal payload. It is given by

$$TN = \frac{n_{l\to l}}{n_{l\to l} + n_{l\to w}} \tag{3}$$

- **False Negative** is defined as the ratio of worm payload that were incorrectly classified as normal payload. It is given by

$$FN = \frac{n_{w\to l}}{n_{w\to l} + n_{w\to w}} \tag{4}$$

- **Accuracy** is defined as ratio of total number of predictions that are correct. It is given by

$$Ac = \frac{n_{w\to w} + n_{l\to l}}{n_{w\to w} + n_{l\to l} + n_{l\to w} + n_{w\to l}} \tag{5}$$

C. *Results and Discussion*

In this subsection, the performance of each of the available data mining algorithm based on classification techniques in WEKA are discussed. The experiments for these are carried out with full data set for training and 10-fold cross validation for testing purposes. The results presented in the following were acquired on a Intel(R) Core(TM)2 Duo CPU, 2.0 GHz and 3GB of RAM.

The four best classification algorithms in terms of accuracy and processing time are chosen–Naive Bayes, J48, SMO, and Winnow. We fixed the result for features that have IG values equal to or more than 0.1 since this data gave the better performance in terms of accuracy and processing time for all classifiers. Table I shows best four classifiers with their corresponding accuracy for ten-fold cross-validation and training set. Comparative duration taken for each technique to classify data are also shown. The detection speed depends to the complexity of these classifiers algorithms.

TABLE I
CLASSIFICATION ALGORITHMS WITH THEIR CORRESPONDING ACCURACY AND PROCESSING TIME

Classifiers	Accuracy (%)		Time
	10-fold C-V	Training Set	
NaiveBayes	94	94	1x
J48	99	99	2x
SMO	99	99	3x
Winnow	99	97	1x

Table I shows the classifiers that could classify dataset for training and ten-fold cross validation with accuracy of between 94% and 99%. J48, SMO and Winnow classifiers produce higher accuracy of classification compared to Naive Bayes classifier. For the processing time, the classifiers time are given as comparative time compared to the shortest time. J48 is two times slower than the Naive Bayes classifier.

The processing time increases when the numbers of dataset increase. Naive Bayes and Winnow classifiers, show good speed performance.

VI. Worm Classification in the Presence of Class Noise

Simulation results in Section V show that content classification techniques using data mining algorithms could be used to detect worms with high accuracy. However, the generative models generation assumes learning corpora with zero class noise. In this section, the robustness of the data mining approach in the presence of class noise is analyzed.

A. *Learning in the Presence of Class Noise*

In this subsection, we observe four classification technique-Naive Bayes, J48, Winnow, and SMO with noisy learning corpora.

Naive Bayes

Based on the Naive Bayes theorem, the posterior probability of a document d_i being in class c_j is

$$P(c_j|d_i) = \frac{P(c_j)P(d_i|c_j)}{\sum_{k=0}^{|all\ classes|} P(c_k)P(d_i|c_k)} \tag{6}$$

$P(c_j)$ in (6) could be easily estimated from the learning data set where

$$P(c_j) = \frac{N_j}{N} \tag{7}$$

In (7), N is the total number of documents in the training set meanwhile N_j is the number of documents labelled with class c_j, for all classes. However, the estimation of likelihood probability $P(d_i|c_j)$ depends on the Naive Bayes model used. Based on the multi-variate Bernoulli model [26], the probability of d_i occurring in trained class c_j, $P(d_i|c_j)$ is defined as

$$\begin{aligned} P(d_i|c_j) &= \prod_{k=1}^{|V|}[B_{ik}P(x_k|c_j) \\ &\quad +(1-B_{ik})(1-P(x_k|c_j))] \end{aligned} \tag{8}$$

where $B_{ik} \in \{0,1\}$ indicates whether feature x_k (e.g., a character, a word, or a phrase) occurs at least once in document d_i and $|V|$ is the number of features learned from learning corpora. $P(x_k|c_j)$ can be estimated as

$$P(x_k|c_j) = \frac{1+\sum_{i=1}^{|D|} B_{ik}P(c_j|d_i)}{|all\ classes| + \sum_{i=1}^{|D|} P(c_j|d_i)} \tag{9}$$

where $|D|$ represents the number of documents in the learning corpora. Meanwhile, $P(c_j|d_i) \in \{0,1\}$ gives a binary value that indicates whether d_i belongs to class c_j or not. The 1 in the numerator and $|all\ classes|$ in the denominator is to prevent $P(x_k|c_j)$ from equaling zero or unity [27]. Since mislabeled examples (class noise) may exist in learning corpora, the estimation of $P(x_k|c_j)$ in (9) is affected. Hence, affected $P(x_k|c_j)$ estimation affects the posterior probability $P(c_j|d_i)$ in (6).

J48

J48 is an enhanced version of C4.5 [28] where in practice, C4.5 uses one successful method for finding high accuracy hypotheses, based on pruning the rules issued from the tree constructed during the learning phase. Given a set S of cases, J48 first grows an initial tree using the divide-and-conquer algorithm as follows:

- If all the cases in S belong to the same class or S is small, the tree is leaf labeled with the most frequent class in S.
- Otherwise, choose a test based on a single attribute with two or more outcomes. Make this test as the root of the tree with one branch for each outcome of the test, partition S into corresponding subsets S_1, S_2, according to the outcome for each case, and apply the same procedure recursively to each subset.

J48 uses two heuristic criteria to rank possible tests: information gain and the default gain ratio. After the building process, each attribute test along the path from the root to the leaf becomes a rule antecedent (pre-condition) and the classification at the leaf node becomes the rule consequence (post condition).

SMO

Support Vector Machine (SVM) performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories. In a linear case, the output of SVM can be defined as

$$u = \vec{w}.\vec{x} - b, \tag{10}$$

where $\vec{w}$ is the normal vector to the hyperplane, $\vec{x}$ is the input vector and b is the threshold. Linear SVM also can be extended to non-linear SVM as derived below by using a kernel function

$$u = \sum_{i=1}^{N} y_i\alpha_i K(\vec{x}_i, \vec{x}) - b, \tag{11}$$

where N is the number of training examples, y is ± 1 value (refer to positive or negative examples), α is Lagrange multipliers, and K is a kernel function (include polynomial or Gaussion kernels [29]) that measures the similarity or distance the input vector $\vec{x}$ and the stored training vector $\vec{x}_i$. The SMO classifier [7] is an improved training algorithm for SVM. This algorithm implements sequential minimal optimization algorithm that solve the smallest possible optimization problem analytically at every step, therefore, it offers a faster technique compare to SVM.

Winnow

Winnow algorithm [30], [31] only updates the weight vector $(w_1, w_2, ..w_n)$ when a misclassified instance is encountered. For a given instance $(x_1, x_2, ..x_n)$ the algorithm predicts the output y as

$$y = \begin{cases} 1 \;\; if \;\; \sum_{i=1}^{n} w_i x_i > \theta, \\ \\ 0 \;\; otherwise \end{cases}$$

where good bounds are obtained if θ is set to $\frac{n}{2}$. The algorithm updates the weights only when a mistake is made as follows:

- if the algorithm predicts 0 and the correct label is 1 (positive example) then the weights of all active x_i (non-zero) is multipled by α (default α is set to 2)
- if the algorithm predicts 1 and the correct label is 0 (negative example) then the weights of all active x_i (non-zero) is divided by α.

In both cases, the weights of inactive x_i (zero value) remain unchanged. The key feature of Winnow is being *mistake driven*. This makes the algorithm more sensitive to the relationships among the features where its mistake bound grows linearly with the number of relevant features.

VII. Experimenting with Worm Trace Files with Class Noise

Based on Fig. 3, we observe the worm detection accuracy using data mining techniques in presence of class noise (normal and worm) by injecting them to the learning corpora.

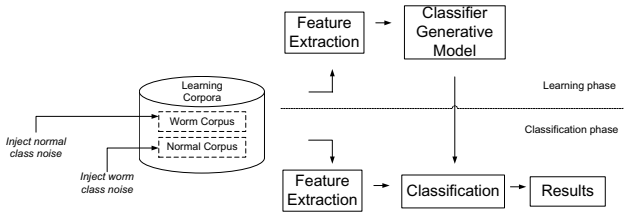

Fig. 3. Worm classification by using data mining techniques in presence of class noise

We used Naive Bayes, J48, SMO and Winnow classifiers from WEKA [7] for this evaluation. We did not modify the classifier and used its full pre-processing functionality in our work since we focus on worm detection accuracy due to the presence of class noise rather than evaluating the effects of classification and learning pre-processing. We use same trace files in the previous work as our learning corpora. Fig. 3 shows the setup similar to Fig. 2. This time we inject the class noise (normal and worm) to the learning corpora. We range both of the class noise injected to the corpus from 0% to 50%. For each level percentage of class noise, we use these four classifiers to classify the data. We evaluate the classification process using ten-fold cross-validation technique.

Table II shows the accuracy of four different classifiers for worm detection in the presence of learning classes noise. The detection accuracy gradually decreases for varying worm class noise and normal class noise between 0% and 50%. For all classifiers, the accuracy drops to about 50% for class noise of 50% for both classes. Beyond the 50% class noise level, we can safely assume that the accuracy will gradually drop to 0%. This trend shows that the presence of classes noise in the learning corpora could not be accepted because it greatly affect the worm detection accuracy.

TABLE II
ACCURACY FOR FOUR DIFFERENT CLASSIFIERS UNDER PRESENCE OF CLASSES NOISE

Classifiers	Classes Noise: Attack \ Normal	0%	10%	20%	30%	40%	50%
J48	0%	0.99	0.96	0.92	0.88	0.84	0.80
	10%	0.94	0.90	0.86	0.81	0.77	0.73
	20%	0.88	0.83	0.78	0.74	0.69	0.66
	30%	0.82	0.77	0.71	0.68	0.64	0.60
	40%	0.76	0.72	0.66	0.63	0.58	0.57
	50%	0.69	0.66	0.60	0.59	0.56	0.57
Naive Bayes	0%	0.94	0.93	0.89	0.86	0.82	0.77
	10%	0.88	0.84	0.80	0.76	0.75	0.70
	20%	0.83	0.78	0.76	0.72	0.71	0.67
	30%	0.81	0.77	0.74	0.69	0.65	0.62
	40%	0.73	0.71	0.68	0.64	0.60	0.59
	50%	0.69	0.65	0.62	0.58	0.59	0.50
Winnow	0%	0.99	0.92	0.89	0.83	0.79	0.75
	10%	0.92	0.85	0.80	0.76	0.72	0.68
	20%	0.86	0.79	0.76	0.71	0.67	0.62
	30%	0.79	0.72	0.70	0.64	0.61	0.56
	40%	0.71	0.63	0.63	0.59	0.55	0.54
	50%	0.66	0.61	0.57	0.54	0.56	0.56
SMO	0%	0.99	0.96	0.93	0.88	0.84	0.80
	10%	0.90	0.86	0.82	0.78	0.74	0.70
	20%	0.81	0.77	0.73	0.69	0.65	0.61
	30%	0.73	0.69	0.64	0.61	0.56	0.52
	40%	0.65	0.60	0.55	0.51	0.49	0.47
	50%	0.55	0.50	0.45	0.46	0.46	0.47

Table III shows the false positive rates of four different classifiers under the presence of classes noise in the learning corpora. The FP for J48 remains under 2% when normal class noise is kept under 10% and worm class noise varies between 0% and 50%. Beyond 10% normal class noise, the FP drops sharply for all worm class noise levels. Naive Bayes is worse than J48 where FP is under 2% for 0% normal class noise, only for worm class noise 30% or less. On the other hand, Winnow and SMO show the class noise more than 10% for both classes will result in high FP, which is unacceptable. This shows that the presence of class noise significantly affects FP.

TABLE III
FALSE POSITIVE FOR FOUR DIFFERENT CLASSIFIERS UNDER PRESENCE OF CLASSES NOISE

Classifiers	Classes Noise: Attack \ Normal	0%	10%	20%	30%	40%	50%
J48	0%	0.003	0.100	0.200	0.300	0.400	0.500
	10%	0.017	0.111	0.202	0.314	0.417	0.539
	20%	0.004	0.115	0.365	0.484	0.595	0.682
	30%	0.000	0.128	0.477	0.575	0.707	0.658
	40%	0.003	0.106	0.303	0.343	0.706	0.888
	50%	0.046	0.109	0.311	1.000	0.912	0.939
Naive Bayes	0%	0.144	0.178	0.246	0.330	0.400	0.500
	10%	0.152	0.260	0.339	0.436	0.430	0.530
	20%	0.158	0.372	0.450	0.418	0.430	0.530
	30%	0.161	0.406	0.487	0.513	0.491	0.530
	40%	0.317	0.399	0.497	0.506	0.537	0.575
	50%	0.236	0.369	0.461	0.439	0.627	0.098
Winnow	0%	0.005	0.095	0.184	0.219	0.302	0.325
	10%	0.093	0.149	0.255	0.286	0.352	0.388
	20%	0.166	0.265	0.337	0.413	0.461	0.461
	30%	0.244	0.394	0.441	0.410	0.596	0.564
	40%	0.420	0.546	0.533	0.624	0.590	0.607
	50%	0.465	0.511	0.597	0.589	0.705	0.592
SMO	0%	0.003	0.100	0.200	0.300	0.400	0.500
	10%	0.106	0.211	0.310	0.408	0.511	0.609
	20%	0.219	0.317	0.431	0.533	0.629	0.729
	30%	0.291	0.410	0.541	0.622	0.734	0.827
	40%	0.377	0.509	0.653	0.737	0.793	0.899
	50%	0.534	0.626	0.769	0.711	0.882	0.899

Table IV shows the false negative rates for four different classifiers under the presence of classes noise in the learning corpora. The FN for all classifiers increase to about 40% with the increases of both class noise. The increases in normal class noise do not significantly affect FN, which has different trend compared to FP as previously discussed. The result shows that SMO gives the best FN. In general, all classifiers show the FN less affected by presence of class noise compared to FP. It can be concluded that the presence of normal class noise in the learning corpora is less likely to affect the detection of worm compared to the presence of worm class noise to the accuracy of worm detection.

TABLE IV
FALSE NEGATIVE FOR FOUR DIFFERENT CLASSIFIERS UNDER PRESENCE OF CLASSES NOISE

Classifiers	Classes Noise: Normal \ Attack	0%	10%	20%	30%	40%	50%
J48	0%	0.002	0.098	0.200	0.300	0.400	0.492
	10%	0.001	0.099	0.200	0.297	0.400	0.506
	20%	0.000	0.100	0.124	0.154	0.357	0.456
	30%	0.000	0.100	0.103	0.144	0.392	0.013
	40%	0.000	0.097	0.107	0.117	0.220	0.110
	50%	0.000	0.092	0.112	0.223	0.118	0.081
Naive Bayes	0%	0.007	0.099	0.184	0.210	0.238	0.367
	10%	0.009	0.096	0.111	0.116	0.219	0.355
	20%	0.009	0.097	0.100	0.107	0.205	0.329
	30%	0.009	0.099	0.186	0.156	0.260	0.405
	40%	0.037	0.125	0.191	0.247	0.306	0.255
	50%	0.051	0.136	0.191	0.285	0.302	0.775
Winnow	0%	0.004	0.076	0.120	0.182	0.197	0.257
	10%	0.063	0.145	0.172	0.206	0.243	0.314
	20%	0.065	0.160	0.170	0.198	0.262	0.318
	30%	0.141	0.214	0.207	0.334	0.272	0.374
	40%	0.144	0.219	0.235	0.248	0.355	0.253
	50%	0.200	0.279	0.319	0.347	0.352	0.338
SMO	0%	0.003	0.092	0.168	0.249	0.339	0.392
	10%	0.002	0.092	0.166	0.240	0.321	0.417
	20%	0.001	0.092	0.162	0.232	0.308	0.392
	30%	0.002	0.092	0.161	0.239	0.316	0.422
	40%	0.001	0.091	0.163	0.231	0.312	0.303
	50%	0.001	0.090	0.164	0.240	0.270	0.274

Our results show that these four classifiers could not tolerate class noise (for worm and normal) classes for worm detection, as presented in Tables II, III, and IV. The presence of class noise in the learning corpora could not be accepted because it greatly affect the worm detection accuracy. As shown in Table III, false positive is more affected by the presence of class noise in worm corpus, which causes classifiers to wrongly identified normal payload as worm payload. However, FN is less affected by presence of normal class noise as shown in Table IV. A general observation from results presented in this paper show that content classification for worm detection must require learning corpora that are free from class noise. Thus, the presence of class noise is not tolerable, even for less than 10% class noise for an accurate worm detection.

VIII. CONCLUSION AND FUTURE WORK

In the first part of the paper, we evaluated the ability of machine learning algorithms to detect worms based on content-based classification. Initial finding from simulation of real packet traces on WEKA show that Naive Bayes, J48, SMO, and Winnow classifiers can detect worms with accuracy between 94% and 99%. J48 produces the best performance than the other classifiers. For the processing time, most classifiers are capable to detect worm with a high speed with the Naive Bayes and Winnow show the best speed performances. The second part of the paper, we analyzed the tolerance of content classification techniques with class noise. By synthetically injecting class noise to the learning corpora, the four best classifiers evaluated in the first part (Naive Bayes, J48, SMO, and Winnow) could not tolerate the presence of class noise greater than 10% especially for worm class learning corpus. Simulation results show worm class noise and normal class noise must be kept below 10% for accurate worm detection.

For future works, we are planning to include more recent and more types of worms. We are also looking forward to implement these algorithms in hardware for real time inline detection. In terms of noise solution, we are planning to use three learning classes; positive, negative and unknown (considered as noise) to analyze the detection accuracy. We also looking forward the potential techniques to reduce effects of noise to the accuracy of worm detection using supervised machine learning techniques.

REFERENCES

[1] J. Rabaiotti, "Malware Detection Using Structural and Behavioural Features and Machine Learning," Ph.D. dissertation, Computer Science, Cardiff University, August 2007.

[2] T. Tafazzoli and S. H. Sadjadi, "Malware fuzzy ontology for semantic web," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 7, pp. 153–161, July 2008.

[3] N. Khiat, Y. Carlinet, and N. Agoulmine, "The Emerging Threat of Peer-to-Peer Worms," in *Proceedings of IEEE/Ist Workshop on Monitoring, Attack Detection and Mitigation*, Tuebingen, Germany, September 2006.

[4] J. Riordan, A. Wespi, and D. Zamboni, "How to Hook Worms," *IEEE Spectrum*, pp. 33–36, May 2005.

[5] A. Schneider, "Methods of Internet Worm Propagation," May 8 2009.

[6] M. Mannan and P. C. V. Oorschot, "On Instant Messaging Worms, Analysis and Countermeasures," in *Proceedings of the 2005 ACM workshop on Rapid Malcode*, Fairfax, VA, USA, November 2005, pp. 2–11.

[7] (2009, July) Data Mining With Open Source Machine Learning Software in Java. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka

[8] E. Z. M. Schultz, E. Eskin and S. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," in *Proceedings of the IEEE Symposium on Security and Privacy*, Los Alamitos,CA, 2001, pp. 38–49.

[9] J. Kolter and M. Maloof, "Learning to Detect Malicious Executables in the Wild," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, USA, August 2004, pp. 470–478.

[10] M. Siddiqui, M. C. WANG, and J. Lee, "Detecting Internet Worms Using Data Mining Techniques," *Journal of Systemics, Cybernetics and Informatics*, vol. 6, no. 6, pp. 48–53, 2009.

[11] W. Wang and D.-S. Luo, "A New Attempt to Detect Polymorphic Worms Based on Semantic Signature and Data-Mining," in *First International Conference of IEEE Communications and Networking (ChinaCom '06)*, Beijing, China, October 2006, pp. 1–3.

[12] M. D. Stopel, "Detecting Malicious Code Activity Based on Computer Behavior Using Artificial Neural Networks," Master's thesis, Ben-Gurion University of the Negev, August 2007.

[13] Y. Yang, Y. Xia, Y. Chi, and R. R. Muntz, "Learning naive Bayes classifier from noisy data," University of California, Los Angeles, CA, USA, Department of Computer Science, Technical Report CSD-TR No. 030056, 2003.

[14] Q. Zhang, "Polymorphic and Metamorphic Malware Detection," Ph.D. dissertation, Computer Science, 2009.

[15] N. Weaver and V. Paxson, "Bayesian Noise Reduction: Contextual Symmetry Logic Utilizing Pattern Consistency Analysis," in *Proceedings of the MIT Spam Conference*, 2005. [Online]. Available: http://bnr.nuclearelephant.com/BNRLNCS.pdf

[16] F. Denis, C. N. Magnan, and L. Ralaivola, "Efficient Learning of Naive Bayes Classifiers Under Class-conditional Classification Noise," in *Proceedings of the 23rd International Conference on Machine learning (ICML'06)*, Pittsburgh, PA, USA, June 2006, pp. 265–272.

[17] C. E. Brodley and M. A. Friedl, "Identifying and Eliminating Mislabeled Training Instances," in *Proceedings of the 13th National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference (AAAI 96/IAAI 96)*, Portland, OR, USA, August 1996, pp. 799–805.

[18] P. Szor, *The Art of Computer Virus Research and Defense*. Addison-Wesley - Symantec Press, 2005.

[19] E. Meister and E. Biermann, "Implementation of a Socially Engineered Worm to Increase Information Security Awareness," in *IEEE Third International Conference on Broadband Communications, Information Technology and Biomedical Applications*, Gauteng, South Africa, Nov 2008, pp. 343–350.

[20] (2009, August) Zdnet Technical Dictionary. [Online]. Available: http://dictionary.zdnet.com/

[21] V. K. T. Abou-Assaleh, N. Cercone and R. Sweidan., "Detection of New Malicious Code Using n-grams Signatures," in *Second Annual Conference on Privacy, Security and Trust*, Fredericton, NB, Canada, 2004, pp. 193–196.

[22] OpenPacket.org. [Online]. Available: http://openpacket.sourceforge.net/

[23] SampleCaptures. [Online]. Available: http://wiki.wireshark.org/SampleCaptures

[24] Protocol Analysis Institute. [Online]. Available: http://www.packet-level.com/traces

[25] Wireshark. [Online]. Available: http://www.wireshark.org

[26] A. McCalum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98): Workshop on Learning for Text Categorization*, Madison, WI, USA, July 1998, pp. 41–48.

[27] L. M. Rudner and T. Liang, "Automated Essay Scoring Using Bayes Theorem," *Journal of Technology, Learning and Assessment*, vol. 1, no. 2, 2002.

[28] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[29] J. C. Platt, "Sequetial Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Microsoft Research, Technical Report MST-TR-98-14, 1998.

[30] N. Littlestone, "Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm," *Journal of Machine Learning*, vol. 2, no. 4, pp. 285–318, 1987.

[31] I. DAGAN, Y. Karov, and D. R. 1997, "Mistake Driven Learning in Text Categorization," in *Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing*, Providence, RI, 1997, pp. 55–63.