On Improving the Accuracy and Performance of Content-Based File Type Identification

Irfan Ahmed[1], Kyung-suk Lhee[1], Hyunjung Shin[2], and ManPyo Hong[1]

[1] Digital Vaccine and Internet Immune System Lab
Graduate School of Information and Communication,
Ajou University, South Korea
{irfan,klhee,mphong}@ajou.ac.kr
[2] Department of Industrial and Information Systems Engineering,
Ajou University, South Korea
shin@ajou.ac.kr

Abstract. Types of files (text, executables, Jpeg images, etc.) can be identified through file extension, magic number, or other header information in the file. However, they are easy to be tampered or corrupted so cannot be trusted as secure ways to identify file types.In the presence of adversaries, analyzing the file content may be a more reliable way to identify file types, but existing approaches of file type analysis still need to be improved in terms of accuracy and speed. Most of them use byte-frequency distribution as a feature in building a representative model of a file type, and apply a distance metric to compare the model with byte-frequency distribution of the file in question. Mahalanobis distance is the most popular distance metric. In this paper, we propose 1) the cosine similarity as a better metric than Mahalanobis distance in terms of classification accuracy, smaller model size, and faster detection rate, and 2) a new type-identification scheme that applies recursive steps to identify types of files. We compare the cosine similarity to Mahalanobis distance using Wei-Hen Li et al.'s single and multi-centroid modeling techniques, which showed 4.8% and 13.10% improvement in classification accuracy (single and multi-centroid respectively). The cosine similarity showed reduction of the model size by about 90% and improvement in the detection speed by 11%. Our proposed type identification scheme showed 37.78% and 31.47% improvement over Wei-Hen Li's single and multi-centroid modeling techniques respectively.

Keywords: file type identification, byte frequency distribution, cosine similarity, Mahalanobis distance, linear discriminant, cluster analysis.

1 Introduction

File type identification is an important task for many security applications to work efficiently. For example, filtering email attachments may require blocking inbound attachment types that may contain malicious contents. Virus scanners may be configured to skip some file extensions; in Norton Antivirus 2008 for

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 44–59, 2009.

example, we have an exclusion option to specify file types to skip for virus scan [1]. Some applications raise alerts before opening unrecognized (therefore suspicious) file extensions; for instance, Windows Media player raises an alert when user tries to open a file of unrecognized extension. Some steganalysis programs also rely on file type detection; for example, stegdetect [2], which detects steganographic contents in images, uses libmagic1 package [3] to determine file type using magic numbers. However, they are easy to be tampered or corrupted so cannot be trusted as secure ways to identify file types. In the presence of adversaries, analyzing the file content may be a more reliable way to identify file types. Solutions that analyze file content usually build a representative model of a file type by averaging the byte frequency distributions of sample files, and use a measure to find out the consistency between the model and the byte frequency distribution of the file in question. Moreover, mahalanobis distance is a popular measure to compare byte frequency distributions such as it is used in PAYL [4], Fileprint [6], [5], [7]. However, existing approaches of file type analysis still need to be improved in terms of accuracy and speed.

In this paper, we propose two schemes to improve the analysis of file content for type identification. First, we show that the cosine similarity [8] is a better metric than mahalanobis distance in terms of classification accuracy, smaller model size and faster detection rate. Secondly, we propose a new type identification scheme that recursively refines the type of a file in question.

We use Wei-Hen Li et al [6]'s modeling technique to compare the cosine similarity (CS) and Mahalanobis distance (MD). Wei-Hen Li's work is a representative modeling technique in that it uses MD and its single centroid model is also used by other schemes [9], [10], [11]. Its multi-centroid model is comparable to our recursive scheme.

In our proposed type identification scheme, we group files with similar byte-frequency pattern irrespective of their file types (whereas Wei-Hen Li's technique groups files with the same type). Then we apply the linear discriminant analysis [12] to identify the type of each file. Our proposed scheme shows better result than Wei-Hen Li's single and multi-centroid modeling technique (considering both CS and MD as distance metric).

This paper is organized as follows. Section 2 presents the related work. Section 3 describes the assumptions that are required for robust techniques of byte-frequency analysis, and explains the dataset used in this paper. Section 4 explains our experiment results on cosine similarity and Mahalanobis distance. Section 5 discusses the weaknesses of Wei-Hen Li's modeling technique. Section 6 presents our recursive file-type identification scheme. Section 7 analyzes the proposed scheme. In section 8, we compare the Wei-Hen Li's single and multi-centroid models (using both CS and MD) with the proposed scheme, and Section 9 shows the conclusions and future work.

2 Related Work

This section discusses the previous techniques for file type identification. Some of these techniques are commonly used by operating systems where the file type

information is explicitly written either in the file header or in the file name. There are other techniques which analyze the file content to identify the file type.

The most common and the simplest way to identify file type is to look at the file's extension [13], but this can be easily spoofed by users with malicious intent. Novice users could also unintentionally change the file extension while renaming the file name. Malwares could also easily hide themselves by having file extensions that virus scanners skip.

Another method to identify file type is to look at magic numbers in the file [14]. For example, GIF files begin with ASCII representation of either GIF87a or GIF89a depending on the standard. ZIP files always have the magic number "PK" at the beginning of the file. However, only binary files have magic numbers so it is not applicable to text files. As with the file extension, it can also be easily spoofed. For instance, malcodes can be hidden using code obfuscation and alphanumeric encoding [15], [16], [17], [18].

McDaniel and Heydari [19] introduce three algorithms to identify file types by analyzing file content. In byte frequency analysis algorithm (BFA), they calculate byte-frequency distribution of different files and generate "fingerprint" of each file type by averaging byte-frequency distribution of their respective files. They also calculate correlation strength as another characterizing factor. They take the difference of the same byte in different files. If the difference gets smaller, the correlation strength increases towards 1 or vice versa. In byte-frequency cross-correlation algorithm, they find the correlation between all byte pairs. They calculate the average frequency between all byte pairs and correlation strength similar to the BFA algorithm. In file header/trailer algorithm, the file headers and trailers are byte patterns that appear in a fixed location at the beginning and end of a file. They maintain an array of 256 for each location and the array entry corresponding to the value of the byte is filled with correlation strength of 1. They construct the fingerprint by averaging the correlation strength of each file. In these algorithms, they compare the file with all the generated fingerprints to identify its file type.

Wei-Hen Li et al. identify file types using n-gram analysis. They calculate 1-gram frequency distribution of files and build 3 different models of each file type: single centroid (one model of each file type), multi-centroid (multiple models of each file type), and exemplar files (set of files of each file type) as centroid. They refer them as "fileprint". In single and multi-centroid models, they calculate mean and standard deviation of 1-gram frequency distribution of files, and use Mahalanobis distance to compare these models with 1-gram distribution of given file to find the closest model. In exemplar file model, they compare 1-gram distribution of exemplar file with that of given file (there is no variance computed), and Manhattan distance is used instead of Mahalanobis distance. Their solution cannot identify files having similar byte-frequency distributions such as MS Office file formats (such as Word and Excel) but treat them as one group or one abstract file type.

Karresand and Shahmehri [9], [10] proposed the *"Oscar"* method for identifying the types of file fragments. They build the single centroid fileprints [6] but use quadratic distance metric and 1-norm as distance metric to compare the centroid with the byte frequency-distribution of file. Although Oscar identifies any file type, they optimized their algorithm for JPG file using specific byte pairs in the file, and reported 99.2% detection rate with no false positives. They also use rate of change of bytes, i.e. the difference of two consecutive byte values where they consider the ordering information of bytes.

Veenman [11] extracts three features from file content. These features are 1) byte frequency distribution, 2) entropy derived from byte-frequency distribution of files, and 3) the algorithmic or Kolmogorov complexity that exploits the substring order [20]. The Fisher linear discriminant is applied to these features to identify the file type.

Calhoun and Coles [21] extended Veenman's work by building classification models (based on the ASCII frequency, entropy, and other statistics) and apply linear discriminant to identify file types. They also argued that files of same type probably have longer substrings in common than that of different types. Our recursive scheme also uses the byte-frequency distribution as a feature and linear discriminant analysis for classification. However, the main difference of ours from Veenman's scheme is the way we build the classification model for each file type. Veenman computes one discriminant function for each file type using all its sample files. However, our scheme combines the similar byte frequency files in groups irrespective of their file types using clustering, and computes the linear discriminant function for each file type in each group. Hence multiple functions could be computed for each file type. Veenman reported 45% overall accuracy while our scheme shows 77% overall accuracy.

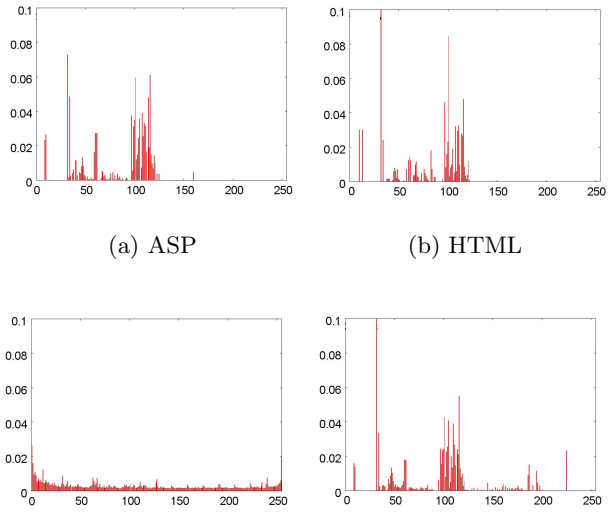
3 Assumptions and Dataset Details

3.1 Assumptions for Byte-Frequency Distributions of File Types

We make two assumptions for byte-frequency distribution of files.

1. Byte-frequency distributions of different file types could be homogeneous. That is, they could have similar byte frequency distributions (Figure 1(a) and 1(b))
2. Byte-frequency distributions of a file type could be heterogeneous. That is, files of the same type could have different byte-frequency distributions (Figure 1(c) and 1(d).

We believe that a robust solution for file-type identification, which uses byte-frequency distribution to build representative models of file types, should satisfy these assumptions. The proposed recursive scheme is based on these assumptions.



(a) ASP (b) HTML

(c) DOC (d) DOC

Fig. 1. a) and b) show that ASP and HTML can have similar byte-frequency patterns. c) and d) show that two DOC files can have different byte-frequency patterns (X-axis represents the byte patterns and y-axis the relative byte frequency).

3.2 Dataset

In our experiments, we used two datasets to analyze different file-type identification schemes and our scheme. One dataset is used to build the representative models of file types and the other to test the classification accuracy of the identification schemes. 10 file types (JPG, HTML, GIF, EXE, MP3, PDF, TXT, DOC, XLS, and ASP) are included in each dataset, with each type having 100 files. These file types are chosen since they are popularly used and cover a broad range of file contents such as binary files (JPG, GIF, EXE, MP3, PDF), printable character files (TXT, HTML, ASP), and the files that contain binary and characters (DOC and XLS).

4 Cosine Similarity: An Efficient Measure to Compare Byte Frequency Vectors

4.1 Cosine Similarity

Cosine similarity is a measure to compare two vectors. If is a byte-frequency vector of a test file and is the representative model vector of a file type (that is achieved by averaging the byte-frequency distributions of the sample files of the file type), cosine similarity is defined by Eq. 1.

$$cos(x, \bar{y}) = \frac{x.\bar{y}}{|x||\bar{y}|} \tag{1}$$

Where . indicates the vector dot product, $x.\bar{y} = \sum_{k=1}^{n} x_k \bar{y}_k$ and $|x| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x.x}$ is the length of vector , The value of cosine similarity lies between 0 and 1. If the cosine similarity is 1, cosine angle between x and $\bar{y}$ is 0, thus x and $\bar{y}$ is the same except the magnitude. If the cosine similarity is 0, cosine angle between x and $\bar{y}$ is 90°, which means they are absolutely dissimilar to each other.

Cosine similarity is quite different from Mahalanobis distance, as it does not use standard deviation but it takes account of the varying size of files. Whereas the byte-frequency distribution of a file needs to be normalized (by dividing each frequency with file size) in order to use Mahalanobis distance, there is no need to normalize the files when using cosine similarity.

In our experimental results presented in next section we found that, unlike Mahalanobis distance, cosine similarity does not lose accuracy even if we truncate the representative model of a file type (so that it contains a subset of highly occurring byte patterns). Thus, using a small number of byte patterns makes it more efficient as it requires less memory and computation without losing accuracy.

4.2 Performance Comparison between Cosine Similarity and Mahalanobis Distance

We use Wei-Hen Li's modeling technique in comparing the classification accuracy of cosine similarity and Mahalanobis distance. The reasons we use their technique are that it uses Mahalanobis distance, its single-centroid model is a representative technique also used by other schemes [9], [10], [11], and its multi-centroid model is comparable to our recursive scheme presented in later section. They build representative model of file type, by combining the byte-frequency distribution of several files of the same type (considering each byte pattern in the distribution as a variable) and averaging their relative frequencies. It is referred as single-centroid model. They also proposed to build multiple representative models of a file type in a similar fashion, but in this case they further group the files of a same type based on their byte-frequency distributions. Such models are referred as multi-centroid model.

We applied, for each file type, multiple truncated models that contain only subsets of byte patterns. For this, we first sorted the byte patterns of a file type (in the order of highest occurring to lowest) and then create multiple models by gradually reducing the number of byte patterns. The rationale behind this is that byte-patterns that rarely occur in a file type may not be representative of that type, but rather be noises. Hence excluding them from the model may be beneficial since it requires smaller memory and computation, and may even increase the classification accuracy in some cases

We also use the simplified Mahalanobis distance formula that is used in Wei-Hen Li's work, which is defined as follows.

$$d(x, \bar{y}) = \frac{\sum_{i=0}^{n-1} (|x_i - .\bar{y}_i|)}{\bar{\delta}_i} \tag{2}$$

where x is a byte-frequency vector of a test file, and $\bar{y}$ and $\bar{\delta}$ are the mean value and standard deviation of the byte-frequency distributions of the sample files (used to build the model of a file type).

Figure 2 (from a to d) shows the classification accuracy of Mahalanobis distance and cosine similarity (on some of the 10 sample file types as mentioned in section 3.2) using different percentages of byte-patterns of single- and multi-centroid models). We used K-means algorithm (as used in Wei-Hen Li's work) to build multi-centroid models. We chose a random value of K (3 in this experiment) as Wei-Hen Li et al. reported that they observed similar results on different values of K. We observed that, in most of the file types, the classification accuracy is degraded as we decrease the number of byte-patterns in the model when using Mahalanobis distance. However using cosine similarity, the classification accuracy either remains the same or shows some improvement.

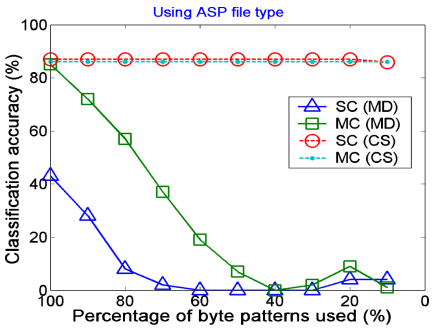
Figure 2 (e) shows the average classification accuracy of Mahalanobis distance and cosine similarity. We can observe that cosine similarity shows better accuracy than Mahalanobis distance, and the multi-centroid model shows better accuracy than single-centroid model. We can also observe that the cosine similarity shows not much difference in classification accuracy even if only 10% of (high frequency) byte-patterns are used. This experiment suggests that we may use a small percentage of high frequency byte-patterns, which reduces the model size and also improve the computation time to compute the cosine similarity values.

Figure 2 (f) shows that by using only 10% of byte-patterns the elapsed time is reduced by approximately 11%. Our experiment considers each byte-pattern as a variable (that is, 1-gram analysis). Since the size of model using 1-gram frequency distribution is small (there are only 256 patterns), there is not much opportunity for improvement on memory space and time. However, models using higher n-grams [22], [23] (i.e. a variable consists of a sequence of multiple bytes) are much bigger since the number of distinct patterns grows exponentially. Therefore the improvement in model size and computation time will be much greater if we use higher n-gram (higher n-gram is out of the scope of this paper).

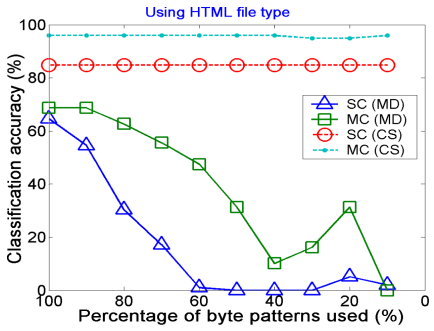
5 Accuracy of Wei-Hen Li et al.'s Single- and Multi-centroid Models

Figure 2 (e) shows that the average classification accuracy using Wei-Hen Li's modeling technique is less than 80%, which may not be satisfactory for many applications. This section presents a discussion of why we think it is so. Our reasoning below is the basis of our proposed scheme in the next section.

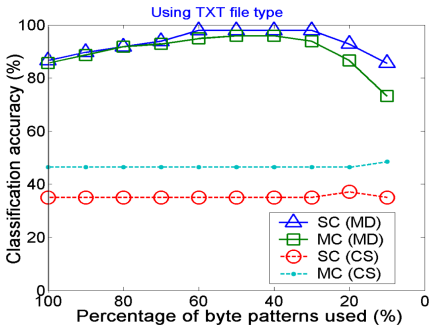
In section 3, we argue that files of a same type could have different byte-frequency patterns. In such case, averaging the different byte-frequency patterns i.e. building a single-centroid model would not yield an accurate representative model (figure 3). In multi-centroid model, a model is built using the files of the same type having similar byte frequency patterns. In section 3, we also argue that files of different types could have similar byte-frequency patterns. Hence it is possible that similar models of different file types are built (figure 4).



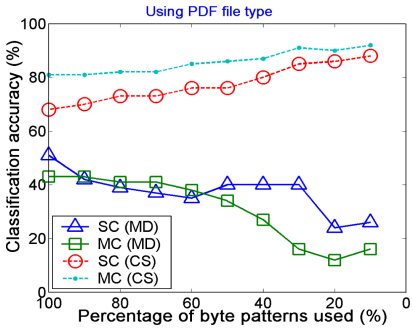
(a) Comparison of MD and CS on SC and MC models using ASP files



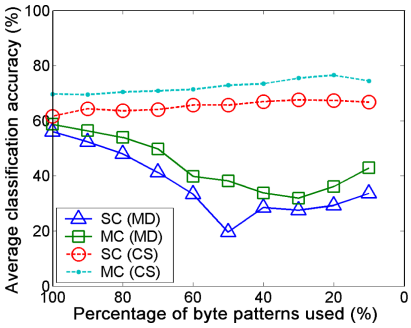
(b) Comparison of MD and CS on SC and MC models using HTML files



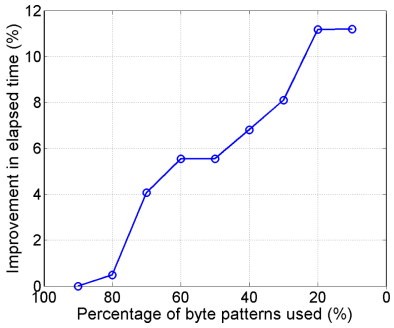
(c) Comparison of MD and CS on SC and MC models using TXT files



(d) Comparison of MD and CS on SC and MC models using PDF files

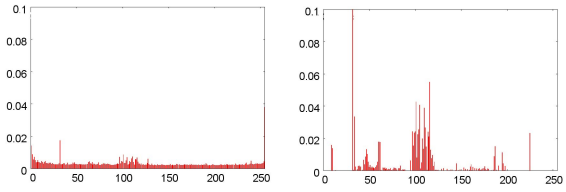


(e) Average classification accuracy of MD and CS on SC and MC models using 10 file types mentioned in figures (a to j)



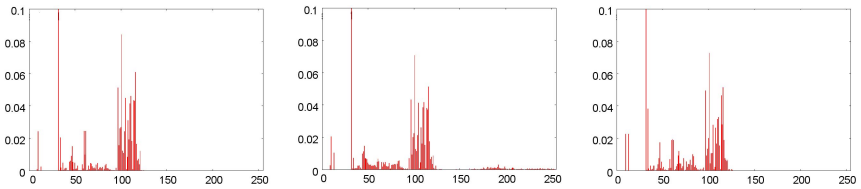
(f) Improvement in elapsed time of file type detection by processing 1500 files (419MB of total size) on different percentages of byte patterns of representative models.

Fig. 2. a) and b) show that ASP and HTML can have similar byte-frequency patterns. c) and d) show that two DOC files can have different byte-frequency patterns (X-axis represents the byte patterns and y-axis the relative byte frequency).



(a) DOC single centroid model (b) A particular DOC file

Fig. 3. Inaccurate single centroid model of the DOC file type. (x-axis represents byte patterns and y-axis is the relative byte frequency).



(a) HTML (b) TXT (c) ASP

Fig. 4. Similar multi-centroid models of different file types. (x-axis represents byte-patterns and y-axis is the relative byte frequency).

Therefore, when identifying the type of a file, both algorithms (i.e. inaccurate single-centroid model and similar multi-centroid models of different file types) are easily confused with similar nature of other file types (such as TXT, ASP, and HTML) and produces inaccurate results.

6 The Proposed File-Type Identification Scheme

A weakness in Wei-Hen Li's multi-centroid model is that it does not cope well with different types of files having similar byte-frequency distributions. Based on this observation, we propose a scheme that applies recursive steps to classify file types. Our scheme builds a representative model in two steps; we first divide files of varying types (in the sample space) into a few clusters, each of which contain files with similar byte-frequency patterns irrespective of their actual types. Then we apply linear discriminant analysis to find the distinguishing byte-patterns (i.e. a model of a file type) from similar byte-frequency distributions in each cluster.

6.1 Building a Representative Model of a File Type

Clustering is an exploratory statistical procedure to naturally group the data into different clusters. At this stage we do not make any consideration on file types.

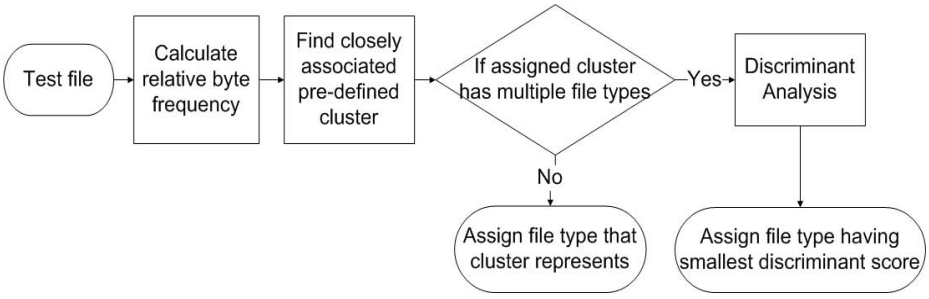


Fig. 5. Identifying a test file using our proposed scheme

After the clusters are built, each cluster could have files with many different types (although the files have similar byte-pattern frequencies). Therefore we need to further divide the type of files in each cluster. For this we perform linear discriminant analysis (LDA), which finds linear combination of byte-patterns to make further distinction of file types. It derives discriminant function for each file type in each cluster. For instance, if two clusters have mp3 files, LDA derives two linear discriminant functions (i.e., one for each cluster). The output of linear discriminant function is called discriminant score, which is supposed to be the least for actual file type.

6.2 Detection Algorithm

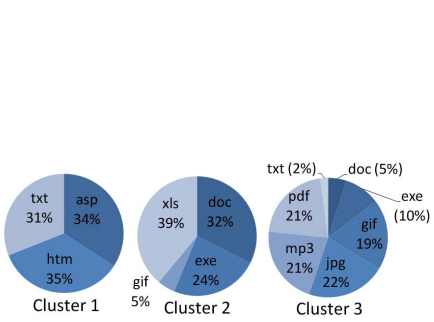
Figure 5 illustrates the detection process of our proposed scheme. When a file enters into the proposed system for type identification, its relative byte frequencies is calculated. The file is then assigned to the cluster having the most similar byte-pattern frequencies. If the assigned cluster represents only one file type, the same type is assigned to the file. Otherwise LDA is used to identify the exact type of the file. The discriminant scores are computed for all file types in the cluster using their respective discriminant function. The file type having least discriminant score is deemed to be the type of file.

7 Evaluation of Our Scheme

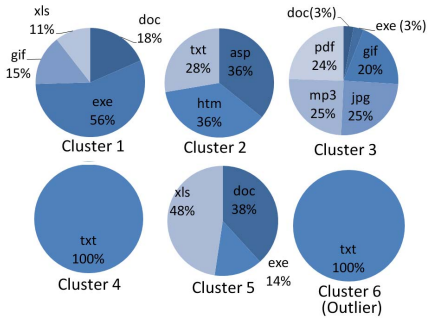
7.1 Cluster Analysis to Group Similar Byte-Frequency Distribution Files

In this section, we describe how files are grouped using clustering. We use Ward's clustering method [24] which forms clusters by minimizing the total within-cluster sum of squares. For instance, if XY is the cluster obtained by combining clusters X and Y, the sum of within-cluster distances (W_{XY}) are

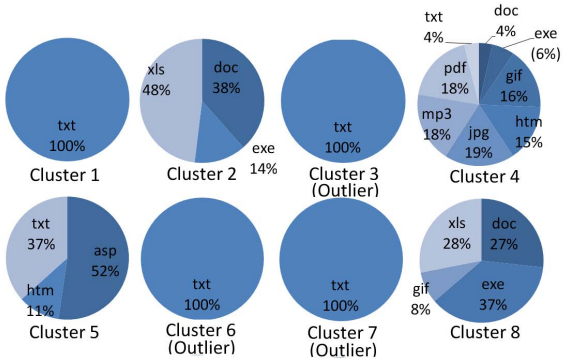
$$W_{XY} = \sum_{i=1}^{n_{XY}} (y_i - \bar{y}_{XY})' (y_i - \bar{y}_{XY}) \tag{3}$$



(a) Grouping file types using 3 clusters



(b) Grouping file types using 6 clusters



(c) Grouping file types using 8 clusters

Fig. 6. grouping of file types on different number of clusters

where y_i are data points, $\bar{y}_{XY} = \frac{(n_X \bar{y}_X + n_Y \bar{y}_Y)}{(n_X + n_Y)}$ and , n_X , n_Y and $n_{XY} = n_X + n_Y$ are number of data points in X,Y and XY respectively.

Figure 6 illustrates the grouping of different file types and their respective percentage. We group files in 3, 6 and 8 clusters. Figure 6 shows that certain file types are always grouped together. These file types usually have similar content characteristics. For example, ASP, HTML, and TXT files usually contain ASCII characters therefore they are grouped together. Also multiple file types lie in one cluster and same file type in multiple clusters. Such observation supports our assumptions made in section 3, which states that files with different types could have similar byte-frequency patterns and files with the same type could have dissimilar patterns.

We observed some outlier clusters and did not use them in type identification process. We consider them as outlier because each of them only consists of one TXT type file.

7.2 Linear Discriminant Analysis to Classify Each File Type

This section discusses the classification accuracy. In detection phase, the test dataset is used to find out the detection accuracy of the clustering, LDA and overall system. If the type of a test file does not match with any of the cluster file type, it is said to be misclassified. For example, if file f of type X is assigned to cluster Y but cluster Y does not have type X, then file f is considered misclassified. Hence the accuracy (%) of assigning files to their respective clusters is calculated as follows.

$$Accuracy(\%) = \frac{(Total\ number\ of\ files\ -\ misclassified\ files)}{total\ number\ of\ files} \tag{4}$$

The files that are not misclassified in the clustering step are further processed by LDA if the assigned cluster has multiple file type. Each cluster has its own linear discriminant functions for its file types. For example, if the file f is assigned

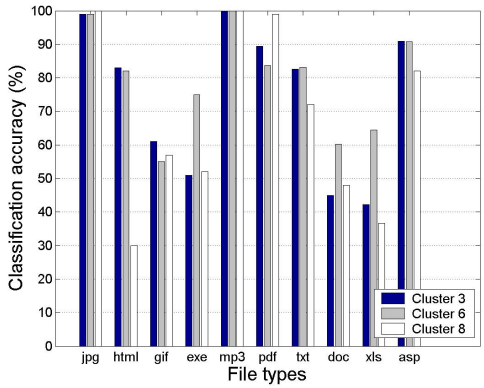


Fig. 7. Accuracy achieved by LDA while detecting 10 file types in different number clusters

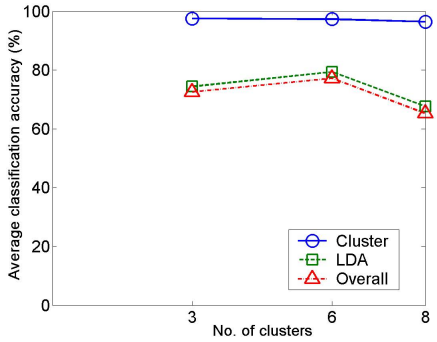


Fig. 8. Average accuracy (%) achieved by clustering and LDA when different number of clusters are used. Overall shows the combined accuracy of clusters and LDA.

to a cluster, linear discriminant score is computed for each file type using its respective function and the file type having least discriminant score is considered as the type of test file f. We use the Eq. 4 to compute the accuracy of LDA for each file type. Figure 7 shows that file types (such as MP3 and JPG) that reside only in one cluster have almost achieved 100% accuracy. However types EXE, DOC, XLS, and GIF reside in multiple clusters (because their byte-pattern frequencies are quite similar) and the results of LDA on them were less accurate. Figure 8 shows the average detection accuracy of the clustering, LDA, and the overall proposed system. Figure 8 also shows the average accuracy of Ward's clustering method when different number of clusters is used. It shows that the accuracy remains almost constant when we increase the number of clusters.

8 Comparison of Single- and Multi-centroid Models with the Proposed Scheme

Figure 9 shows the comparison of the proposed scheme with single- and multi-centroid models (using both cosine similarity and Mahalanobis distance). We

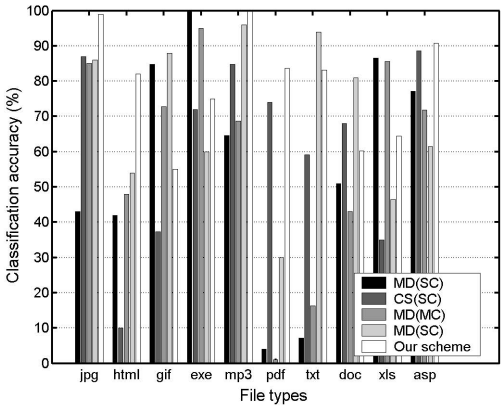


Fig. 9. Classification accuracy (%) achieved by our proposed scheme (6 clusters are used) and single and multiple centroid models using cosine similarity (CS) and Mahalanobis distance (MD) while classifying 10 sampled file types

	Single centroid		Multi centroid		Our proposed scheme (6 clusters)
	MD	CS	MD	CS	
Average accuracy	56.03%	61.59%	58.72%	69.66%	77.20%
Improvement by proposed scheme	37.78%	25.35%	31.47%	10.82%	

Fig. 10. Comparison of average classification accuracy among single- and multi-centroid models and the proposed scheme

took the classification accuracy using all the 256 byte-patterns in single and multi-centroid models (i.e. we did not truncate the models). Figure 9 shows that in most cases our recursive scheme has highest accuracy rate, and on average it achieved substantial improvement in classification accuracy over single- and multi-centroid models (Figure 10).

9 Conclusion and Future Work

This paper presented two approaches to improve the classification accuracy in identifying file types using their byte-frequency distributions. The first approach is to use cosine similarity as a distance metric to find out the consistency between the byte frequency-distribution of a test file and representative model of a file type. Based on our experimental results, we conclude that cosine similarity is a better measure than Mahalanobis distance as it improved 4.8% and 13.10% classification accuracy using Wei-Hen Li's single and multi centroid models respectively, and did not lose accuracy when using a small percentage of highly-occurred byte-patterns in the representative models. Using small percentage of byte patterns results in smaller model size and faster detection rate. Our experiment shows that cosine similarity reduces the detection speed by approximately 11% when only 10% of highly-occurred byte-patterns of the representative model are used. We expect that the improvement in model size and computation time will be much greater if we use higher n-gram.

Wei-Hen Li's multi-centroid model performs better than its single-centroid model but still suffers when building models of different file types showing similar byte patterns. We proposed a recursive scheme which is comparable to Wei-Hen Li's multi-centroid model but is more accurate (figure 10 shows 31.47% improvement).

From cluster analysis, we noticed that files of similar nature file types are always grouped together. For instance, the text-based files (ASP, TXT and HTML) are always grouped together. Also DOC and XLS which contains both text and binary characters are always found in same clusters. The accuracy of linear discriminant in our experiment is still not satisfactory. As our future work, we plan to improve our grouping scheme, especially focusing on identifying text-based file types.

Acknowledgement

This research is supported by the Ubiquitous Computing and Network(UCN) Project, Knowledge and Economy Frontier R&D Program of the Ministry of Knowledge Economy(MKE) in Korea and a result of subproject UCN 09C1-C5-20S.

H.Shin would like to gratefully acknowledge support from Post Brain Korea 21 and the research grant from Korean Government (MOEHRD, Basic Research Promotion Fund, KRF-2008-531-D00032).

References

1. Exclusion option to skip the files for the scanning in Norton antivirus, http://service1.symantec.com/SUPPORT/nav.nsf/0/c829006aa01d540b852565a6007770d8?OpenDocument
2. Stegdetect, http://packages.debian.org/unstable/utils/stegdetect
3. Libmagic1 package, http://packages.debian.org/unstable/libs/libmagic1
4. Wang, K., Stolfo, S.J.: Anomalous Payload-based Network Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004)
5. Ahmed, I., Lhee, K.-s.: Detection of malcodes by packet classification. In: Workshop on Privacy and Security by means of Artificial Intelligence, ARES 2008, pp. 1028–1035 (2008)
6. Li, W.J., Wang, K., Stolfo, S., Herzog, B.: Fileprints: Identifying File Types by n-gram Analysis. In: Workshop on Information Assurance and security (IAW 2005), United States Military Academy, West Point, NY, pp. 64–71 (2005)
7. Srinivasan, N., Vaidehil, V.: Reduction of False Alarm Rate in Detecting Network Anomaly using Mahalanobis Distance and Similarity Measure. In: Proceedings of ICSCN, pp. 366–371 (2007)
8. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to data mining. Addison-Wesley, Reading (2005)
9. Martin, K., Nahid, S.: Oscar - file type identification of binary data in disk clusters and RAM pages. In: IFIP security and privacy in dynamic environments, pp. 413–424 (2006)
10. Martin, K., Nahid, S.: File type identification of data fragments by their binary structure. In: Proceedings of the IEEE workshop on information assurance, pp. 140–147 (2006)
11. Veenman, C.J.: Statistical disk cluster classification for file carving. In: IEEE third international symposium on information assurance and security, pp. 393–398 (2007)
12. Rencher, A.C.: Methods of Multivariate Analysis. Wiley Interscience, Hoboken (2002)
13. File extensions, http://www.file-extension.com/
14. Magic numbers, http://qdn.qnx.com/support/docs/qnx4/utils/m/magic.html
15. Nachenberg, C.: Polymorphic virus detection module, United States Patent # 5,826,013 (1998)
16. Szor, P., Ferrie, P.: Hunting for metamorphic. In: Proceedings of Virus Bulletin Conference, pp. 123–144 (2001)
17. RIX, Writing IA32 Alphanumeric Shell codes, http://www.phrack.org/issues.html?issue=57&id=15#article
18. Eller, R.: Bypassing MSB Data Filters for Buffer Overflow Exploits on Intel platforms (2003), http://community.core-di.com/~juliano/bypassmsb.txt
19. McDaniel, M., Hossain Heydari, M.: Content Based File Type Detection Algorithms. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (2003)
20. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Problems of Information Transmission, 1–11 (1965)

21. Calhoun, W.C., Coles, D.: Predicting the types of file fragments. Digital Investigation 5(1), 14–20 (2008)
22. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 226–248. Springer, Heidelberg (2006)
23. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation: in 16th USENIX Security Symposium (2007)
24. Ward, J.H.: Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association, 235–244 (1963)