A Comparative Study of File-Type Identification Techniques

Nasser S. Alamri
Computer Science and Cybersecurity
Florida Institute of Technology
Melbourne, FL, USA
nalamri2005@my.fit.edu

William H. Allen
Computer Science and Cybersecurity
Florida Institute of Technology
Melbourne, FL, USA
wallen@fit.edu

***Abstract*— Research in file-type identification has employed a number of different approaches to classify unknown files according to their actual file type. However, due to the lack of implementation details in much of the published research and the use of private datasets for many of those projects, it is often not possible to compare new techniques with the prior work.**

In this paper, we present a comparison of five common file-type identification approaches, along with the parameters used to perform the comparisons. All approaches were evaluated with the same dataset which was drawn from public or widely-available sources. Our results show that each approach can produce good results with 88% to 97% classification rates, but achieving these results requires "tuning" the parameters of the inputs to the classifiers.

Keywords—digital forensics, cybersecurity, machine learning, feature extraction, file-type identification

I. INTRODUCTION

The identification of a file's type (e.g., image, text document, spreadsheet, executable program, web page, etc.) is an important problem in both digital forensics and cybersecurity. Applications rely on protocol or file format specifications to correctly access the header fields and data in files. However, to process an input file, the application must first be able to determine the correct file type. In the past, it was sufficient to rely on a file's name extension or on information in the file's header, such as a magic number or text string, to identify a file's specific type. Recent changes in attack patterns show that these approaches are no longer sufficient to avoid misidentifying harmful files. Attackers are aware that it is possible to change a filename extension to one that is associated with a "safe" file type or to craft a false header that displays the correct magic number or text string, but is not indicative of the file's true type or contents. Forensic investigators have often found critical evidence in files that were modified to hide the fact that they contain images or other information that is associated with criminal activities [9].

Several classification techniques have been developed to detect file types. However, no techniques have yet been proven to be capable of universal identification, nor has any one approach been successful with all known file formats. Thus, file type identification is still an open problem that deserves further study [12]. Unfortunately, much of the existing research made use of private datasets that are not available to use for comparison with new approaches. The comparison of classification techniques will clearly be improved by use of an open dataset that is large enough to provide a wide range of file types and with sufficient diversity of sources to produce a "real-world" evaluation of the proposed classification schemes.

The goal of this project is to devise a method for comparing recent file-type identification approaches to evaluate their classification rate with different types of files. The majority of published research in this area used data that was not publically available and was often not clearly described. As a result, it is not possible to directly compare the performance of different approaches with each other or with new techniques that are focused on this problem. Some researchers did their own comparison between different approaches with their datasets, but the results were not consistent from one research project to another. As a result, one publication may present evidence that a specific approach performed better than another approach (on their data) and a different publication may show results that contradict that outcome.

Garfinkel [11] has collected a very large set of files taken from U.S. Government websites and other public sources and made it available to forensics researchers for a variety of purposes. Because the bulk of the data comes from government sources, it is commonly referred to as the GovDocs data [11]. Acceptance of the GovDocs data for digital forensics research is increasing and a few of the more recent file-type identification research projects (e.g., [7][10]) did use files selected from that dataset. We believe that the use of large numbers of files selected randomly from the GovDocs data will enable researchers to more accurately compare the classification rates of file type identification techniques.

II. RELATED WORK

We surveyed the recent literature on file-type identification and identified the most commonly used approaches [4]. From that study, we selected five of the most successful techniques to use for our comparative study. In this section, we briefly describe each of the five approaches, along with references to the researchers using those approaches. It should be noted that

The authors would like to thank the Institute of Public Administration (IPA) in Saudi Arabia for their support of this research.

978-1-4673-7300-5/15/$31.00 ©2015 IEEE

the majority of the published research projects used different datasets to evaluate the author's approaches, making comparisons between projects difficult. As stated, our goal was to overcome this limitation by using a common dataset for all experiments. Therefore, we chose to select files from the publicly-available GovDocs dataset described above [11].

A Support Vector Machine (SVM) is a supervised learning technique used for classifying and recognizing data. It projects data into a high dimensional space where the data is classified based on generating hyper-planes that separate the data in those high dimensions [14].

A number of recent research projects have used an SVM for classifying file types by analyzing the file's content. Ahmed, et al. [2] compared an SVM to several other approaches, including a Neural Network, k-Means clustering and Linear Discriminant Analysis, and reported similar results for all of those techniques. Amirani, et al. [6] used an SVM classifier to evaluate a group of different types of files by extracting major features from each file and using those features to classify the file's type. They compared the SVM's results against a Neural Network classifier and found that the SVM performed better than the Neural Network classifier. Beebe, et al. [7] created a new technique for identifying file types by concatenating unigrams (single bytes) and bigrams (byte pairs) to create n-grams (groups of bytes) that are used as input to an SVM. They performed experiments with a classifier SVM that used either a linear kernel or a Radial Base Function (RBF) kernel and found that the RBF kernel was slower and did not scale well with larger sample sizes compared to the linear kernel [7].

Other researchers used an SVM. For example, Fitzgerald, et al. [10] worked with file fragments of several different sizes and explored the use of an SVM with a linear kernel. They found that the classification accuracy increased only slightly with larger fragments but that it performed better than several similar classification approaches. Gopal, et al. [13] evaluated the performance of existing Commercial Off-The-Shelf (COTS) software for file type identification of corrupted files and then compared those results to the classifications produced by techniques such as an SVM. They reported that the SVM technique using bigrams as input data performed better than the COTS software. Sportiello, et al. [16] decomposed files into 512 byte blocks with an average of 28,000 blocks per file type. They used several common statistical measurements, such as entropy and byte frequency, to produce input for the SVM classifier with the goals of increasing the classification rate while decreasing the error rate.

The K-Nearest Neighbor (k-NN) [14] approach attempts to classify test data by determining which particular class of data the test data is closest to. Ahmed, et al. [2] compared a k-NN classifier to several other classifiers to determine how much time was required to process the contents of each file and which technique was most accurate. Among all techniques they used, they found that the k-NN approach outperformed the others. Gopal et al. [13] reported that using bigrams as input to a k-NN technique outperformed the same approach with unigrams as input. They found that the k-NN approach produced similar results to the SVM technique.

The Neural Network (NN) is a well-known technique used in several areas of computer science for classifying data or detecting patterns that can be used for prediction [14]. Ahmed, et al. [3] proposed applying a Neural Network to analyze a file's content and divided this approach into three major phases. In the first phase, the cosine similarity measure is applied to extract byte frequencies from a number of data files. In the second phase, groups of similar files are used to build a model for each file type. In the third phase, a Multi-Layer Perceptron (MLP) classifier is implemented to classify the files by type.

To reduce the input to a Neural Network to the most important features, Amirani, et al, [5] used Principal Component Analysis (PCA) to select the features that would best represent a file's content. When they used a Multi-Layer Perceptron (MLP) classifier to process the features extracted by using PCA, they achieved a better accuracy rate than with other classifiers [5]. Penrose et al. [15] used a Neural Network for file type identification and focused on compressed and encrypted files. Furthermore, different fragment sizes were used and they achieved an overall correct detection rate between 70% and 75%.

Linear Discriminant Analysis (LDA) is another dimensional reduction technique with a goal that is similar to PCA in that it transforms the contents of a large data file type to a smaller set of meaningful features from the original data file [14]. However, LDA can also be used to as a classifier because it can find a linear combination of features which can be used to distinguish between two or more classes of data. Ahmed, et al. [1] compared the classification of data files using LDA with classifying the same types of files with a Cosine-Similarity technique. The classification accuracy rate of the techniques presented in [1] were not as good as those in several of the other papers, with an accuracy of 77.2%. Calhoun and Coles [8] applied LDA to the problem of identifying a file type from file fragments, particularly those found in the middle of the data file. They found that LDA performed well when a large number of file fragments were available.

III. METHODOLOGY

Research in file-type identification has employed a number of different approaches to classify unknown files according to their actual file type [4]. Some approaches to file-type identification have found modest results with simple techniques, such as byte frequency, without regard to file structure. Other methods create a 'fingerprint' or signature for each file type and classify an unknown file by matching its signature to one of the known file types. Despite the success described in a number of recent publications, the lack of implementation details in much of the published research and the use of private datasets for many of those projects means that it is not possible to compare new techniques with much of the recently published work.

Additionally, the published results from comparing existing techniques are often contradictory, with one researcher showing that technique A is better than B and another showing that B produces better results. We believe that two causes of these inconsistent results are the choice of limited, unverifiable

datasets and the lack of published information regarding the experimental parameters used to evaluate the approaches being compared.

In this paper, we address this problem by performing a comparative study of five common file-type identification approaches drawn from the literature (see Table I). To assist with future comparisons, we performed our experiments with randomly-selected sample data from the well-documented public source known as the "GovDocs" dataset [21].

The file types chosen for this study include PDF documents, JPG images, ASCII text files (TXT), Microsoft Word documents (DOC) and HTML pages. We also included executable files from the standard installation disk for Windows 7. We selected at least 150 files of each type to use for the experiments and then randomly selected subsets of those files, as described below.

TABLE I. CLASSIFICATION TECHNIQUES SELECTED FOR COMPARISON

Classifier for File Type Identification
Support Vector Machine
k-Nearest Neighbor
Neural Network with Radial Basis Function kernel
Neural Network with Multilayer Perceptron kernel
Linear Discriminant Analysis

A. Data and Parameter Selection for the Experiments

Table II shows the four configuration parameters that determined the nature of the input data to the classification process. Because many of the classifiers perform better with a limited number of features as input data [6], we used Principal Component Analysis (PCA) to extract a specific number of features that were used to create the models used for classification. Table II shows the range of feature counts that were used as input to the classifiers and the tables in the next section show which features counts produced the best results.

Another parameter that had an impact on the classification results was the overall number of files used for evaluating each technique. Acknowledging that using fewer files will reduce processing time, we tested each approach with as few as 30 files but found that the results were consistently better with at least 50 to 100 files, therefore none of the results tables in the next section show 30 files as a parameter. On the other hand, some approaches performed well with as few as 50 files, as shown in Tables III through VII. It should be noted that the 30, 50 and 100-file samples were selected randomly from a larger collection of files that had been extracted from the GovDocs or Window 7 files described in Section I.

File fragment classification is an increasingly important approach for determining the contents of a file. It is based on the idea that dividing a file into smaller fragments produces input data that is more easily processed than an entire file. Therefore, we evaluated the use of five different fragment sizes for each input file, ranging from 500 bytes to the entire file. In each case where we used less than the entire file, we extracted the file fragment from the beginning of the file. Our results showed that larger fragments (including using the entire file) produced consistently poorer results, thus they do not show in the tables in the next section.

When constructing our classification models, we tried different ratios of training data to test data. The ratios shown in Table II and in the Results section refer to the percentage of training data / the percentage of test data. To produce training and test data, we partitioned the input files selected for the experiments by randomly choosing some for training data and some for test data, in the proportion specified in Table II.

To evaluate the parameters that we had determined could have an impact on the performance of a classifier, we tested all combinations of the parameters and analyzed the results to find the best parameter combinations for each classifier we used.

TABLE II. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE LINEAR DISCRIMINATE ANALYSIS APPROACH

Parameter for Experiment	**Range of parameters used**
Number of features input to the classifier	2, 4, 8, 12, 16, 32, or 64
Number of files in the sample set used as input	30, 50, 100
Size (in bytes) of file fragment used as input	500, 1000, 2000, 8000, whole file
Ratio of training data to test data	50/50, 60/40, 70/30, 80/20, 90/10

B. Experimental Process

Files were randomly selected from the datasets described in Section I and preprocessed to prepare them for use as training or test data. As explained above, file fragments were extracted and PCA was used to select the correct number of features for each parameter combination. The sets of extracted features were divided into training data and test data and the training data was used to create a model that would be used for classification of the test data. Each combination of parameters (number of features, fragment size, ratio of training data and file count) was used to generate a separate model. After the training data was used to create a model for each classifier, the test data was evaluated and the results were determined.

The results from each of the combinations of the parameters were analyzed and the best outcomes for each classification approach are shown in Tables III through VII in the next Section. Due to space limitations, we do not show all of the results from our evaluations. Table VIII summarizes the results by showing the best results for each classifier and the set of parameters that produced that result.

IV. RESULTS

Although our experiments used all combinations of the parameters listed in Table II, we show only the combinations that produced the best results for each classifier in Tables III through VII. In each case, the Principal Component Analysis technique was used to extract the number of features specified in the table from the file fragments. The Training Data Ratio column refers to the ratio of the training data used to create the

classification model to the test data used for classification of the input file. Although a 60/40 ratio was also used in the experiments, it did not produce better results than the ratios shown in the tables and was omitted due to space limitations.

Similarly, we tested 8000-byte fragments and also extracted features from whole (non-fragmented) files, but the results from those data sizes were not as good as the results from the 500, 1000 and 2000-byte fragments, so they do not appear in the tables.

Table VIII shows the best combination of parameters for each of the classification approaches. In the next section, we discuss the results and present our conclusions.

TABLE III. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE LINEAR DISCRIMINATE ANALYSIS APPROACH

LDA	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	64	500 byte	90/10	93%
100 files	64	500 byte	80/20	88%
50 files	64	1000 byte	70/30	91%
100 files	64	1000 byte	80/20	93%
50 files	64	2000 byte	70/30	80%
100 files	64	2000 byte	80/20	85%

TABLE IV. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE SUPPORT VECTOR MACHINE APPROACH

SVM	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	64	500 byte	90/10	93%
100 files	64	500 byte	80/20	94%
50 files	16	1000 byte	90/10	93%
100 files	64	1000 byte	80/20	92%
50 files	16	2000 byte	90/10	90%
100 files	64	2000 byte	80/20	92%

TABLE V. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE K-NEAREST NEIGHBOR APPROACH

k-NN	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	32	500 byte	80/20	90%
100 files	8	500 byte	90/10	97%
50 files	16	1000 byte	70/30	84%
100 files	12	1000 byte	90/10	95%
50 files	16	2000 byte	90/10	90%
100 files	12	2000 byte	70/30	89%

TABLE VI. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE NEURAL NETWORK (RBF) APPROACH

NN-RBF	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	4	500 byte	80/20	82%
100 files	4	500 byte	80/20	87%
50 files	4	1000 byte	90/10	80%
100 files	4	1000 byte	80/20	88%
50 files	4	2000 byte	80/20	77%
100 files	4	2000 byte	80/20	88%

TABLE VII. PARAMETERS THAT PRODUCED THE BEST CLASSIFICATION RATE FOR THE NEURAL NETWORK (MLP) APPROACH

NN-MLP	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
50 files	64	500 byte	90/10	97%
100 files	32	500 byte	70/30	92%
50 files	32	1000 byte	90/10	93%
100 files	64	1000 byte	80/20	94%
50 files	16	2000 byte	70/30	87%
100 files	32	2000 byte	80/20	91%

TABLE VIII. BEST RESULTS FOR EACH APPROACH

	Files	Number of Features	Fragment Size	Training Data Ratio	Classification Rate
LDA	50	64	500 byte	90/10	93%
SVM	100	64	500 byte	80/20	94%
k-NN	100	8	500 byte	90/10	97%
NN-RBF	100	4	1000 byte	80/20	88%
NN-MLP	50	64	500 byte	90/10	97%

V. DISCUSSION

In this paper, we presented a comparison of five common file-type identification approaches, along with the parameters used to perform the comparisons. All approaches were evaluated with files selected from a dataset which was drawn from public or widely-available sources [11]. Our results show that each approach can produce good results with 88% to 97% classification rates, but achieving these results requires "tuning" the parameters of the inputs to the classifiers.

As mentioned in Section I, one of our goals was to determine whether fragment size, training data ratio and number of features could be tuned to improve the classification rate. The results from these experiments were mixed, with no particular combination of parameters producing consistently good results across all classifiers. As shown in Table VIII, each classifier worked best with a unique set of parameters.

VI. Future Work

Using this comparison as a basis for future work, we intend to evaluate new classification approaches, improve the classification of individual types of data and try different feature selection algorithms. For example, as shown in [4], published file identification techniques use either PCA or LDA for feature extraction. However a significant body of research has addressed similar problems in other fields, such as image processing, biometric authentication and data mining, and has proven that other data reduction approaches are useful for feature extraction. Therefore, we will explore the application of feature extraction algorithms from those fields to the file identification problem and will also evaluate the use of classifiers that have proven successful in those domains. Further work is also needed to optimize the parameters for classifiers to match the different file types and sizes.

References

[1] Ahmed, Irfan Kyung-Suk Lhee, Hyun-jung Shin, and Man-Pyo Hong. "On Improving the Accuracy and Performance of Content-Based File Type Identification." In Information Security and Privacy: 14th Australasian Conference, ACISP 2009, July 1-3, Proceedings, vol. 5594, p. 44. Springer, 2009.

[2] Ahmed, Irfan, Kyung-Suk Lhee, Hyun-Jung Shin, and Man-Pyo Hong. "Fast content-based file type identification." In Advances in Digital Forensics VII, pp. 65-75. Springer Berlin Heidelberg, 2011.

[3] Ahmed, Irfan, Kyung-Suk Lhee, Hyun-jung Shin, and Man-Pyo Hong. "Content-based file-type identification using cosine similarity and a divide-and-conquer approach." IETE Technical Review 27, no. 6 (2010): 465.

[4] Alamri, Nasser S., and William H. Allen. "A taxonomy of file-type identification techniques." In Proceedings of the 2014 ACM Southeast Regional Conference. ACM, 2014.

[5] Amirani, Mehdi Chehel, Mohsen Toorani, and A. Beheshti. "A new approach to content-based file type detection." In Computers and Communications, 2008. ISCC 2008. IEEE Symposium on, pp. 1103-1108. IEEE, 2008.

[6] Amirani, Mehdi Chehel, Mohsen Toorani, and Sara Mihandoost. "Feature‐based Type Identification of File Fragments." Security and Communication Networks 6, no. 1 (2013): 115-128.

[7] Beebe, N., L. Maddox, Lishu Liu, and Minghe Sun. "Sceadan: Using Concatenated N-Gram Vectors for Improved File and Data Type Classification." (2013): 1-1.

[8] Calhoun, William C., and Drue Coles. "Predicting the types of file fragments.", Digital Investigation 5 (2008): S14-S20.

[9] Conti, Gregory, Erik Dean, Matthew Sinda and Benjamin Sangster, "Visual Reverse Engineering of Binary and Data Files", in Proceedings of the 5th international workshop on Visualization for Computer Security, 2008

[10] Fitzgerald, Simran, George Mathews, Colin Morris, and Oles Zhulyn. "Using NLP techniques for file fragment classification." Digital Investigation 9 (2012): S44-S49.

[11] Garfinkel, S. "Bringing science to digital forensics with standardized forensic corpora", Digital Investigation, Vol. 6, 2009, pgs 2-11.

[12] Garfinkel, S. Lessons learned writing digital forensics tools and managing a 30TB digital evidence corpus, Digital Investigation, Vol. 9, pg 80-89, 2012

[13] Gopal, Siddharth, Yiming Yang, Konstantin Salomatin, and Jaime Carbonell. "Statistical learning for file-type identification." In Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on, vol. 1, pp. 68-73. IEEE, 2011.

[14] Kulkarni, Sanjeev, and Gilbert Harman. An elementary introduction to statistical learning theory. Vol. 853. John Wiley & Sons, 2011.

[15] Penrose, Philip, Richard Macfarlane, and William J. Buchanan. "Approaches to the classification of high entropy file fragments." Digital Investigation (2013).

[16] Sportiello, Luigi, and Stefano Zanero. "Context-Based File Block Classification." In Advances in Digital Forensics VIII, pp. 67-82. Springer Berlin Heidelberg, 2012.