Third International Symposium on Information Assurance and Security

Statistical Disk Cluster Classification for File Carving

Cor J. Veenman[1,2]

[1]Intelligent System Lab, Computer Science Institute, University of Amsterdam, Amsterdam
[2]Digital Technology and Biometrics Department, Netherlands Forensic Institute, The Hague

Abstract

File carving is the process of recovering files from a disk without the help of a file system. In forensics, it is a helpful tool in finding hidden or recently removed disk content. Known signatures in file headers and footers are especially useful in carving such files out, that is, from header until footer. However, this approach assumes that file clusters remain in order. In case of file fragmentation, file clusters can be disconnected and the order can even be disrupted such that straighforward carving will fail. In this paper, we focus on methods for classifying clusters into file types by using the statistics of the clusters. By not exploiting the possible embedded signatures, we generate evidence from a different source that can be integrated lateron. We propose a set of characteristic features and use statistical pattern recognition to learn a supervised classification model for a range of relevant file types. We exploit the statistics of a restricted number of neighboring clusters (context) to improve classification performance. In the experiments we show that the proposed features indeed enable the differentation of clusters into file types. Moreover, for some file types the incorporation of cluster context improves the recognition performance significantly.

1. Introduction

In computer forensics an important problem is recovering files from (part of) digital media for which no file system information available [10]. The file system information can be missing for several reasons. The file of interest may have been deleted such that the file system indexes no longer refer to the file content. In that case the file content is usually unchanged until the clusters are overwritten with other files. On the other hand, the files may be contained on a device with an unknown file system. Last, the files may be hidden from the file system for instance by masking parts of the medium out as unavailable and putting probably compromising content there. From a forensic perspective revealing these covert files is clearly relevant, since they may be deleted or hidden exactly because they are compromising for the owner of the media.

The process of file recovery from block or cluster based media is known as file carving. This problem is receiving increasingly scientific attention. Since a few years a research challenge is organised to let the scientific community objectively compare their results[1]. One of the main problems with file carving is file fragmentation. If files were never fragmented, discovering the start of a file and the end of it would be enough to completely restore the file. Discovering the first and the last cluster of a file is rather easy for many file types, because signatures are used to indicate headers and sometimes footers and even intermediate clusters. In case of file fragmentation file clusters become disconnected and can be out of order. Moreover, for several file types no useful header signature information is available.

In this paper, we follow another approach. We attempt to establish the file type of an individual cluster from which it originates. In order to achieve this, we use the statistical content of the clusters of the respective file types. We do not exploit signatures that can be available in file headers and footers of certain file types. As such this approach delivers an additional source of information that can be used in the file carving process.

On a file basis rather than on cluster basis, recognition of file type is also done for malware detection [3], [8], [9], [12]. The aim is then to detect if files have malicious embedded content or just a modified file extension. An important difference with file carving is that files can be considered as a complete unit of information. Either because the file system can be used or the network protocol in case in-

[1]http://www.dfrws.org/

0-7695-2876-7/07 $25.00 © 2007 IEEE
DOI 10.1109/IAS.2007.75

coming files are scanned for their content. In any case larger (and complete) amounts of data can be used for the analysis, while headers are always available if they are defined for the specific file type.

In the next section, Section 2, we first transform the cluster type recognition problem into several classification problems. We propose a set of features to describe the content of a cluster and turn the classification problems into supervised learning problems. In the experiments section, Section 3, we describe the dataset with available file types that we use for learning and testing. We introduce the notion of cluster context and show how it helps in improving classification performance.

2 Statistical Learning

In the process of revealing hidden file cluster information, several recognition problems could be stated. First, as an aid in carving files out for which a header has been detected, recognizing clusters belonging to the same file is helpful. To this end, a model should be learned that exploits the difference between within-file variation and between-file variation generalized over file types. When a substantial part of a file already has been recovered, the clusters from the recovered file part can help in establishing a specialized model or for tuning parameters of a model. Alternatively, specific models could be learned to screen a disk image for clusters of a given type with *two-class* classification or detection models. However, when screening the disk for one file type after the other, clusters may be labeled as several different file types because of false positives of the respective models. A way to prevent these conflicting labels, is by posing the file type recognition problem as an exclusive *multi-class* classification problem. Because in such a problem file types have to be decided exclusively, it shows clearly which file types are confused with each other.

In this research, we will deal with both the two-class file type recognition (which we call here *type-x* recognition) problem and the multi-class file type classification problem (which we call *type-all* recognition).

It should be noted that with a recognition problem in this domain, there are some fundamental problems. These problems originate from the fact that several file types allow for the embedding of other file types, such as images in word processing documents. Accordingly, without (the possibility of) parsing the files it is impossible to establish whether or not a file is self-contained or embedded. For these clusters the file type is truely ambiguous.

Cluster Features

In order to typify the content of a cluster one could use all the bits in the cluster as features. However, to be a significant feature for a certain file type the feature should be invariant to transformations that are normal for 'any' file content. Since we want to decribe and differentiate all kinds of files, the features cannot be fitted to describe invariants of specific file types, but instead general file content *invariants*, such as:

- some chunks of data have no specific order, e.g. functions in programming language source files
- chunks of data can be reordered without becoming completely different, e.g. uncompressed images
- data can be shifted while remaining syntactically correct and semantically similar, e.g. by entering line feeds in text files

Alternatively, several file characteristics are specific to certain file types and therefore help in differentiating these from other files. We call these characteristics file type *variants*, such as:

- data can have specific order
- certain symbols appear far more frequent, e.g. '$<$' and '$>$' in html files
- files usually contain redundant information, but the degree of redundancy differs greatly between files, e.g. ZIP versus HTML

Based on these rationales, we use the following cluster content features:

- The histogram of byte values. The histogram only accounts for byte frequencies and ignores the byte order. The (slightly transformed) histogram has been used as so-called "fingerprint" for malware detection in [8] and [9]. It has also been used similar to our application, i.e. for byte fragments on disk or in memory [5].
- The information content measured as entropy [11]. The entropy is derived from the byte histogram and therefore also ignores the byte order. Because the entropy is computed through a *non-linear* function from the histogram it can be a useful feature, especially when linear recognition models are used (see next section).
- The algorithmic or Kolmogorov complexity. In contrast with the entropy, this measure exploits substring order [6]. We use the algorithmic complexity as defined in [7]. Before computing the algorithmic complexity we transform the byte stream into several classes in order to make more sense, such as ASCII digit, white space, bracket, other printables, and non ASCII (> 128).

Supervised Learning

For classifying data sequences to a certain file type, we use a statistical pattern recognition approach. We apply supervised learning on a labeled dataset for which we know for all clusters the true file type. After computing the set of features for every cluster, we learn a model that is able to predict the class label based on the features. Both for the two-class and multi-class configuration, we use the Fisher classifier, see e.g [2]. We select this classifier for its accuracy and its computational efficiency.

Also in [5], [8] and [9], a prediction model is based on labeled training data. The main difference with our approach is that the model is mainly designed rather than learned from the data. In all these works, a prototypic histogram is build per file type through a form of averaging. Further, some distance measure is used to compute the closest prototypic histogram. Then, the predicted file type is the one from the closest prototype. In [5], a threshold is addded that imposes a maximum on the distance to the prototypes. When the distance of a cluster to the closest prototype exceeds the threshold, then it will not be assigned to that file type but as rest type instead.

3 Experiments

For the *type-x* and *type-all* recognition we used a dataset that we collected from the internet. The dataset consists of 11 file types in total 450 MB with file extensions AVI, BMP, DBX, DOC, EXE, GIF, HTML, JPG, PDF, PPT and ZIP. We did sanity checks, though not thoroughly, to verify the file extension with the file content. Per class the dataset contains three to twenty thousand clusters of 4096 bytes with known class (file type) labels. The file types in the dataset represent more or less what can be expected and what are possibly interesting file types from a forensic perspective, such as images, movies, documents and email archives.

Type-all recognition

First, we did the *type-all* experiments. In these experiments the goal is to classify all clusters into one of the pre-defined file types. In order to learn a suitable model and to evaluate it, we randomly separated the dataset into one third for training and two third for testing, where the training part is used to learn the classification model. Accordingly, we had a training set of approximately $35,000$ clusters and a test set of approximately $70,000$ clusters. The separation of the dataset in training and test data is done on a file basis. That is, from a file all clusters are in the training set or in the test set. It is important to divide the data strictly, because usually clusters from the same file are similar. Accordingly, the estimated performance would become too optimistic if clusters from one file are in both the training and the test data.

After applying the learned Fisher classifier to the test data we obtain the confusion matrix as displayed in Table 1. The confusion matrix shows what file types are assigned to clusters from the test set using a model learned on the training data. In that way, it can be seen which file types are confused most and which file types are more easy to recognize.

From the confusion matrix it can be concluded that given the proposed features, several file types can be recognized quite well while others are hard to distinguish. Especially, almost all HTML and JPG clusters are correctly recognized as such. It can, however, also be seen that several cluster types are classified as being JPG which are in fact not, see the column JPG in Table 1. This also holds for HTML clusters, but to a lesser extent (column HTLML in the same table). Further, ZIP clusters are strongly confused with JPG and PPT clusters and also several cluster types are incorrectly recognized as ZIP, such as GIF and PPT (see column ZIP in Table 1).

The overall accuracy is 0.45 which is quite modest. It is, however, a substantial improvement over the prior probabilities of the respective file types. That is, with 11 classes the prior error is $1 - 1/11 = 0.91$.

Type-x recognition

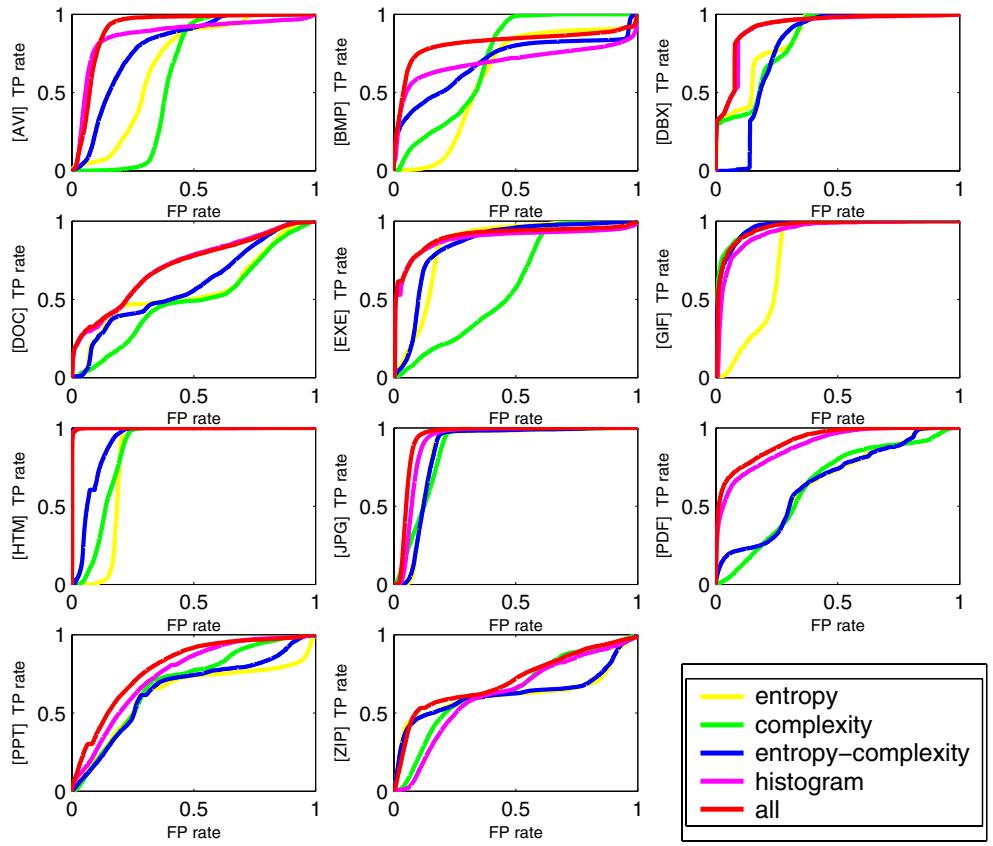
The next problem we attacked is the *type-x* recognition problem. Again we separated the dataset into one third for the training set and two third for the test set on a file basis. In turn we learned a classifier on the training data where we set one file type as target type and all others as rest type. For these problems, we use the ROC curve as performance measure [2]. The ROC curve has the advantage over the classification accuracy, that it shows how false negatives and false positives are balanced. More precisely, in the ROC curve the true positive (TP) rate is plotted against the false postive (FP) rate, where positives are the clusters from the target file type. In Fig. 1, we show the ROC curve for all type-x problems. In order to see the added value of the proposed features, we plotted separate curves for entropy, complexity, histogram, and all features in the respective figures.

The figures show that when all features are exploited, especially clusters from HTML files are easy to recognize. It has to be noted that in this data set no programming language source files or other XML type files are present, which makes it probably easier to detect these HTML file clusters. The figures show that especially DOC and ZIP clusters are much harder to find. In order to find these file types a substantial amount of false recognitions have to be accounted for.

In order to quantitatively measure the difference in per-

	AVI	BMP	DBX	DOC	EXE	GIF	HTML	JPG	PDF	PPT	ZIP
AVI	0.56	0.01	0.01	0.04	0.07	0.01	0.00	0.14	0.03	0.07	0.06
BMP	0.20	0.36	0.00	0.03	0.18	0.00	0.03	0.05	0.04	0.05	0.05
DBX	0.00	0.00	0.45	0.01	0.23	0.00	0.21	0.01	0.08	0.01	0.00
DOC	0.01	0.06	0.01	0.39	0.07	0.00	0.02	0.33	0.00	0.09	0.03
EXE	0.02	0.04	0.02	0.04	0.78	0.00	0.01	0.03	0.01	0.03	0.03
GIF	0.00	0.00	0.00	0.00	0.00	0.35	0.00	0.08	0.02	0.35	0.20
HTML	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00
JPG	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.01	0.00
PDF	0.01	0.00	0.01	0.00	0.01	0.01	0.00	0.18	0.59	0.10	0.08
PPT	0.00	0.00	0.00	0.04	0.05	0.00	0.00	0.36	0.01	0.38	0.15
ZIP	0.00	0.00	0.02	0.02	0.06	0.00	0.03	0.39	0.07	0.22	0.18

Table 1. Confusion matrix resulting from classifying the test set using a multi-class model learned on the training set. Each row shows into which file types clusters of a certain file type are classified. In other words, the left column denotes the true file type labels and the top row denotes the predicted file type labels. For perfect performance the diagonal of the matrix should contain 1 and all other entries 0.



(a) ROC (b)

Figure 1. Receiver Operating Characteristic (ROC) performance curves for all file types. The figures show per file type the performance with entropy feature, algorithmic complexity feature, both entropy and complexity features, histogram feature, and all features.

.	.	$i-2$	$i-1$	i	$i+1$	$i+2$	.	.
.	.	.	.	0	.	.	.	.
.	.	.	1	1	1	.	.	.
.	.	2	2	2	2	2	.	.

Figure 2. Illustration of cluster contexts. Each cell represents one cluster. The top row shows cluster with order number i and its surrounding clusters. The other rows show which clusters are involved in the context for cluster i with context size $0, 1$ and 2, respectively.

formance for different feature sets, we used a derived scalar measure from the ROC curve, that is, the Area Under the ROC Curve (AUC) [1]. The AUC has an interesting statistcial interpretation. It expresses the probabilty that a random target class object will have a higher relevance for the target class than a random rest class object [4]. So, still no decision threshold is assumed.

In Table 2, we show the mean AUC averaged over 20 runs for all file types and several feature subsets. In Fig. 1 and Table 2, it can be seen that the histogram is the most important feature. However, for AVI, BMP and DBX the cluster entropy and algorithmic complexity also clearly add to the recognition rate and are therefore useful features.

Recognition with added cluster context

A final experiment we did to improve the recognition performance was by adding the context of a cluster. With cluster context we mean preceeding and consecutive clusters on the disk, see Fig. 2. Accordingly, the effective size of the cluster increases, such that more reliable statistics can be computed. The disadvantage is that sometimes clusters of other file type are used to compute features. These will disrupt the characteristics and, hence, the recognition rate. Without fragmentation this problem will only arise at the beginning and end of a file. With fragmentation clearly more clusters will suffer from *negative* context.

In Table 3, we show the AUC performance results for the type-x experiments with increasing context size. Especially, the recognition of DOC clusters strongly favors from the added context. There are, however, also some clusters that can be recognized more difficult when neighboring clusters are incorporated in their statistics through context, such as BMP and JPG clusters. The size of the files is not the cause of this problem (for instance the smallest BMP file is 300 kB).

4 Conclusion

In this paper, we dealt with the classification of file clusters on digital media into several known file types. File cluster classification can serve as an intermediate step in a forensic file carving application. We defined a set of relevant features and a supervised learning scheme to learn recognition models, both for a direct multi-class classification problem as well as several per file type two-class classification problems. We ran multi-class and two-class recognition experiments using a dataset of 450 MB that we collected from the internet. It seems that the two-class (or type-x) recognition models are more useful, since the false positive rate can be adjusted per file type. The multi-class (type-all) recognition models have, however, the advantage that they clearly show which file types are confused with which ones. Although, the cause for these confusions is not fully clear, focusing on them helps in defining more distinctive features.

In these experiments we also incorporated surrounding cluster data (context) in order to increase the effective cluster size for feature computation. From the experiments, we can conclude that clusters of several file types can be recognized with a low false positive rate. Accordingly, for these file types the proposed methodology can be helpful in the file carving process. However, further study is needed to intelligently integrate these recognition results with classifications obtained by detection of signatures. An unsolved problem in this respect is the ordering of possibly correctly classified clusters (in terms of file type).

In general there is room for improvement of the proposed method. In experiments not reported here, it already became clear that with increased training set size the performance improves. However, especially the memory requirements will become a limiting factor for the used Fisher classifier. Furthermore, other features can be considered, while on the other hand the histogram may be compressed to reduce the data need for training. Finally, although the performance increase by using cluster context is limited, it may be worthwhile to apply. However, the influence of fragmentation should be studied to see what is the best trade-off between context size and expected degree of file fragmentation.

Acknowledgements

We thank Nicolaas Heyning and Wouter Suren for data preparation and feature extraction for the experiments. Further, we thank Dr. Raoul Bhoedjang for proofreading and valuable comments.

features	AVI	BMP	DBX	DOC	EXE	GIF	HTML	JPG	PDF	PPT	ZIP
entropy	0.69	0.60	0.79	0.67	0.82	0.78	0.82	0.85	0.64	0.62	0.68
complexity	0.60	0.71	0.81	0.55	0.61	0.97	0.87	0.84	0.61	0.68	0.72
entr/compl	0.80	0.71	0.79	0.68	0.84	0.94	0.92	0.84	0.62	0.65	0.69
histogram	0.87	0.72	0.91	0.74	0.91	0.97	0.99	0.90	0.88	0.74	0.75
all	0.93	0.74	0.93	0.74	0.90	0.97	0.99	0.90	0.89	0.75	0.75

Table 2. Mean Area Under the ROC Curve (AUC) performance for all file types and feature subsets averaged over 20 runs, that is, 20 draws of one third training set and two third test set.

context	AVI	BMP	DBX	DOC	EXE	GIF	HTML	JPG	PDF	PPT	ZIP
0	0.95	0.81	0.96	0.76	0.94	0.98	1.00	0.91	0.86	0.82	0.70
1	0.95	0.77	0.96	0.80	0.95	0.98	1.00	0.90	0.88	0.82	0.70
2	0.95	0.74	0.97	0.81	0.96	0.98	1.00	0.89	0.88	0.82	0.70
4	0.96	0.72	0.97	0.83	0.96	0.98	1.00	0.88	0.89	0.82	0.71
8	0.96	0.73	0.98	0.85	0.97	0.97	1.00	0.86	0.89	0.82	0.72

Table 3. Area Under the ROC Curve (AUC) performance for all file types with increasing context size. For the DOC clusters there is clear increase in performance, while the BMP clusters clearly suffer from included context in the feature computation.

References

[1] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[2] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., New York, 2001.

[3] R.F. Erbacher and J. Mulholland. Identification and localization of data types within large-scale file systems. In *Second International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE'07)*, pages 55–70. IEEE Computer Society, 2007.

[4] J.A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982.

[5] M. Karresand and N. Shahmehri. Oscar - file type identification of binary data in disk clusters and ram pages. In *Proceedings of the 21st International Information Security Conference (IFIP SEC'06)*, pages 413–424, 2006.

[6] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11, September 1965.

[7] A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Transaction on Information Theory*, 22:75–81, 1976.

[8] W-J. Li, S.J. Stolfo, and B. Herzog. Fileprints: Identifying file types by n-gram analysis. In *Proceedings of the 2005 IEEE Workshop on Information Assurance*, pages 64–71, West Point, NY, June 2005.

[9] M. McDaniel and M. Hossain Heydari. Content based file type detection algorithms. In *Proceedings of the 36th Hawaii International Conference on Systems Sciences (HICCS'03)*, volume 09, page 332a. IEEE Computer Society, 2003.

[10] G.G. Richard and V. Roussev. Next-generation digital forensics. *Communications of the ACM*, 49(2):76–80, 2006.

[11] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July 1948.

[12] S.J. Stolfo, K. Wang, and W-J Li. *Malware Detection*, volume 27, chapter Towards Stealthy Malware Detection, pages 231–249. Springer US, 2007.