UNIVERSITY OF AMSTERDAM

THESIS PLAN

DRAFT V 0.2

Text Fragments Classification in Digital Forensics

Author:

Ioannis D. TZANELLIS

Supervisor:

Jeroen van den BOS

June 29, 2013

Contents

Introduction File recovery from digital data storage devices has been a hot topic among the Digital Forensics field. Traditional data storage devices make use of file systems, in order to manage contained data, their available space and to maintain location of files. When the storage device and its file system are intact, it is quite simple to recover data from them. This is mainly because file systems make use of meta-data in order to track information for their files. Meta-data can contain information such as creation date, data struc- ture (e.g directory or regular file), file type, file owner, size, last modified date etc. In a real life forensic case it is highly unlikely that file meta-data will be present, or they might be corrupted or deleted. It became clear for the digital forensic community that an alternative, more realistic approach must be used.

1 File Carving

File carving is a forensics technique that recovers files based on their content, without relying on their meta-data. File carving process involves two steps. File validation and file reconstruction.[1]. During the recovery procedure, we must first validate the type of the file and then apply the appropriate reconstruction technique. In this thesis, only the validation techniques are of our interest. By examining the byte-content and/or the structure of a file [22], file validation techniques are used to classify its type. Several file types contain common structures like headers, footers (named Magic Number Matching) [7][3], fields that specify file attributes like color or size etc.(Data Dependency Resolving [3]), that can be used to identify the type of the file. Additionally, another approach is to apply statistical analysis techniques and algorithms, which use the complete byte set of a file, creating a fileprint for every file type. Some examples are the n-Gram Analysis [9], the Byte fre- quency analysis (BFA) algorithm and the Byte frequency cross-correlation (BFC)[4]. The aforementioned techniques have some profound weaknesses. The Magic Number Matching and the Data Dependency Resolving approaches make general type classification infeasible. This is due to the fact that not every file-type contain such structures. Furthermore, n-Gram Analysis and both BFA and BFC were designed to be applied in a complete file or a pre-defined part of it, which retains all of its content. Hence, they depend on files internal structure and characteristics.

2 Problem Formulation

So why this is a problem? The answer lies in file systems behaviour and file fragmentation. When we delete a file from a media storage, the data are not actually removed. The sectors in which the file was stored still contain the same data, although the file system marks them as unallocated [2]. Which means that the next time a new file is created, the file system is free to use these sectors, which are marked as unallocated, to store the new file. But if the new file is bigger than the old one, and the file system tries to store it starting from the same sector entry as the deleted one, it wont have enough space to store it. So the file system will allocate all the sectors of the previous deleted file, while the remaining data which do not fit, will be stored to other unallocated sectors. This results to file fragmentation.

In a forensic file recovery case, it is more probable that the files that must be recovered are fragmented. Validation techniques which use the complete file content are highly unlikely to provide aid to forensic examiners. Hence, an alternative approach to file type classification must be taken. File fragment classification is a technique that uses only a small fragment of a file, in order to determine its type. Ergo, file fragmentation is not a problem any more as this approach is independent from files overall structure. Although in theory, file fragment classification looks like an ideal approach, in practice current solutions that use this approach could not yield good results[][]. One reason that file fragment classification is difficult, is due to the complex container files. Complex container files like TAR, ZIP, RAR, PDF etc. contain other primitive file types, making general fragment classification difficult. Moreover, a fragment might contain more data which are strongly related to the files content than the files structure.

3 Objectives

The main objectives in this project are:

1. Test the hypothesis that by analysing only a special ASCII byte-set of file fragments, which corresponds to the printable ASCII characters plus the tab, newline and carriage return character, accuracy of classification algorithms can be enhanced for document-type frag-

ments.

2. Create a more accurate algorithm for identifying document-type fragments than the available ones. In particular text, xls, doc and pdf files are our main focus and we try to improve their classification accuracy.

4 Algorithms Requirements

The design requirements for our classification algorithm are as follows:

1. Speed - Relatively fast compared to current techniques
2. Accuracy - Algorithm should be as accurate as possible by minimizing false positives in classification of file fragments
3. Operate in 512 bytes, same as the sector size of a hard drive

5 Methodology

Most of the current file and fragment classification techniques use the whole byte content of a file/fragment for both the training and classification procedures. Since we intend to create an algorithm which would be able to yield better accuracy results for fragments that originate from a document file type, we want to test the hypothesis that by using only the printable ASCII characters ($32 \geq b \leq 126$) plus the tab, newline and carriage return of a fragment we could achieve better results regarding text fragment classification. The aforementioned special characters are a behavioural trait of a document so we expect that their occurrence in conjunction with all the other printable characters will be more frequent in a document file.

In order to test our hypothesis we have to use one of the current classification algorithms in order to compare their accuracy results. Additionally, since our main goal is to design a classification algorithm which will satisfy the already mentioned requirements, we should carefully choose a currently available algorithm that has the potential to be easily modified, without adding additional complexity, and to create a custom more effective version of it.

Our algorithm of choice is the Byte Frequency Analysis algorithm. More about the reasons of this choice can be found in the chapter 3.

It has been observed that BFA, although extremely inaccurate, classifies a big amount of fragments that belong to a document file as text. We will make use of the BFA which will take under account only our special subset of ASCII characters among with 3 more variations of it and try to enhance its accuracy on classifying document-file fragments as text. We used 4 different variations of the BFA mainly for 2 reason. The first one is to compare the results with the results of the BFA that Shahi used for file fragment classification and find out if our hypothesis is correct. The second one is to choose the variation of it that yields the best results regarding text fragment classification. Literally this is going to be the first step of our algorithm so accuracy of our BFA variation will affect the accuracy of our final algorithm. Next we will isolate all fragments classified as text and analyse them in order to find patterns which will eventually result in special metrics that could help us to design our algorithm.

Related Work

6 Related Work

Here I will put the related work*

Experimental Setup

7 Data Set

The data set we used for both our training and testing procedures is derived from Garfinkels[] coprus, Wikipedia and Academic Earth[] and is the exact same coprus that Shahi[] used for his testing set.The set is comprised of 10 different file types with a size of about 1GB each. We split this corpus in a 9-1 ratio for the training and testing set respectively. Furthermore, we divided both the testing and training set files in 512-byte blocks, which we refer to them as fragments. We used the training set to train our fingerprints and apply statistical analysis in order to discover useful patterns and the testing set to test all variations of our BFA algorithm. More detailed information about our data set can be found in Table **??**.

Table 1: Data Set

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
Training Set										
num.of files	1,642	1	954	1,697	1	373	193	1,781	464	4,395
size in megabytes	869.3	860.6	831.2	867.6	813.6	869.5	866.9	870.5	863.4	868.9
expected num. of fragments	1,780,326	1,762,508	1,702,297	1,776,844	1,666,252	1,780,736	1,775,411	1,782,784	1,768,243	1,779,507
output num.of fragments	1,694,034	1,680,771	1,622,534	1,467,314	1,588,908	1,684,374	1,683,444	1,698,877	1,685,954	1,692,813
percentage of fragments with no plain text	4.8	4.6	4.7	17.4	4.6	5.4	5.2	4.7	4.7	4.9
Testing Set										
num.of files	217	1	367	257	1	81	35	214	101	555
size in megabytes	100	104.9	97.4	100.2	104.9	100.2	100.6	100.2	100.2	101.5
expected num. of fragments	204,800	214,835	199,475	205,209	214,835	205,209	206,028	205,209	205,209	207,872
output num.of fragments	189,732	204,795	190,055	177,887	204,728	193,352	195,289	195,608	195,656	195,653
percentage of fragments with no plain text	7.4	4.7	4.7	13.3	4.7	5.8	5.2	4.7	4.7	5.9

8 Byte Frequency Analysis(BFA) Algorithm

BFA [McDaniels] is a statistical learning algorithm that was initially developed to analyse and classify whole files. It was not meant to be used in file fragments. By counting the number of instances of each byte in a file of a certain type, BFA uses this frequencies to create a representative average value for each byte instance, among with their respective correlation strength. This results in a fingerprint for this particular file-type. Then during the classification procedure, the input file is compared with every fingerprint and an accuracy level is created for each file type. BFA classifies the file to the file type that corresponds to the highest accuracy level. Shahi trained and tested BFA with file fragments of 512-byte size and his results show that although the algorithm is pretty bad for broad fragment classification, it is quite good in identifying fragments that belong to document-type files as text. We use a BFA which will train our fingerprints with the bytes that corresponds to the printable ASCII characters plus the tab, line break and carriage return instead of the complete byte-set of the fragments. This BFA will be only the first step of our algorithm and we intend to use additional metrics after this point. Taking under account the speed requirements, BFA seems as a very good candidate since it is quite a lightweight technique compared to heavier machine learning algorithms. Moreover, as we already stated, BFA seems to classify a big ammount of document-type fragments as text. Shahi tested several classification algorithms in the same corpus. BFA was also tested among with Byte Frequency Correlation algorithm, n-Gram Analysis and Conti et al. algorithm. The results show that BFA has the highest precision in classifying document-type fragments as text.

BFA Variations

9 Variation 1 - Special ASCII subset fingerprint training

In this variation we created 10 fingerprints which were trained with fragments from the training set, one for each file type. We used only the printable ASCII characters ($32 \geq b \leq 126$) among with the tab(9), new line (10) and the carriage return(13) characters. The results can be found in Table **??**.

This variation of BFA classifies 589,758 fragments as text which corresponds to the 30.4% of the initial corpus. 501,012 of them are fragments that come from pdf, xls, doc and text files and 88,746 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 85% of true positives in identifying document-type fragments as text with 15% false positives. This 85% of true positives corresponds to the 66.7% of the total pdf, xls, doc and text files of our corpus.

Table 2: BFA Results - Fingerprints with printable ASCII characters

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	189,732	204,795	190,055	177,887	204,728	193,352	195,608	195,608	195,656	195,653
pdf	27.9	52.3	0.0	20.3	48.1	0.2	35.3	40.7	46.5	44.1
zip	20.2	26.6	0.0	13.3	28.0	0.1	24.9	29.2	24.7	28.2
text	21.3	4.9	98.0	50.4	4.4	95.5	14.1	6.0	7.1	7.2
doc	14.4	4.2	0.5	7.1	5.2	0.2	9.7	7.9	8.7	5.8
mp4	1.7	0.6	0.0	0.2	0.8	0.0	0.4	0.5	0.4	0.5
xls	1.2	0.0	1.4	0.8	0.1	3.9	1.0	0.2	0.0	0.1
ppt	3.2	2.2	0.0	1.8	2.7	0.0	3.3	3.3	2.7	2.9
jpg	0.5	0.1	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.1
ogg	2.8	2.2	0.0	1.4	3.0	0.0	2.8	3.0	2.7	2.7
png	6.8	6.9	0.0	4.6	7.7	0.0	8.3	9.1	7.2	8.5
Unclassified	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

10 Variation 2 - 4-Ratio categories of our special ASCII subset

During our research we thought that it would be interesting to analyse the distribution of bytes that belong to our special ASCII subset of the training set fragments. Depending on the percentage of our special ASCII subset in a fragment, the fragment was assigned to one of 4 ratio categories. 0-25%, 25-50%, 50-75% and 75-100%. The results of this analysis can be found in Table **??**. As it seems fragments from certain file types are more likely to belong to certain ratio categories. For example almost all text fragments(99.95%) contain more than 75% of our special ASCII subset and almost all xls fragments less than 50%. Undoubtedly this is completely reasonable. Text files are mostly comprised of plain text and Excel sheets, with their cell-like structure, contain less printable characters. And this analogy is more obvious in a 512-byte fragment. That finding can be used as a metric to improve current classification techniques and we are going to elaborate more on this later in this document.

Based on the analysis results we thought that would be interesting to divide the fragments of our training set in 4 such categories. Then for each category and for each file type we created their respective fingerprints. So we ended up with 40 fingerprints, 4 for every file type. The algorithm checks first the ratio of our special ASCII subset of the input fragment and according to its value it compares the fragment with the fingerprints of the respective category. The results of this BFA variation can be found in Tables **??**, **??**, **??** and **??**.
The accuracy for both the actual classification and the text classification are really bad. This vari-

Table 3: Training Set Ratio of special ASCII subset Analysis

ratio	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
0 - 25%	9,327	347	235	528,661	3,130	1,054,503	114,968	7,842	785	11,875
25 - 50%	1,332,849	1,680,052	436	768,686	1,585,760	576,755	1,547,585	1,685,320	1,684,877	1,674,301
50 - 75%	86,583	370	181	8,834	18	31,595	10,106	1,305	287	1,787
75 - 100%	265,275	2	1,621,682	161,133	0	21,521	10,785	4,410	5	4,850
Total:	1,694,034	1,680,771	1,622,534	1,467,314	1,588,908	1,684,374	1,683,444	1,698,877	1,685,954	1,692,813
0 - 25%	0.55	0.02	0.01	36.03	0.20	62.61	6.83	0.46	0.05	0.70
25 - 50%	78.68	99.96	0.03	52.39	99.80	34.24	91.93	99.20	99.94	98.91
50 - 75%	5.11	0.02	0.01	0.60	0	1.88	0.60	0.08	0.02	0.11
75 - 100%	15.66	0	99.95	10.98	0	1.28	0.64	0.26	0	0.29

Table 4: BFA - Fingerprints Trained in 0-25% and tested in 0-25%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	5,714	90	3	52,264	2,854	147,873	11,027	1,332	222	7,874
pdf	0	0	0	0	0	0.3	0	0.1	0	0
zip	0	0	0	0	0	0	0	0	0.5	0
text	0	0	0	0.1	0	0.7	0	0	0	0
doc	0	0	0	0	0	0	0	0.1	0	0
mp4	0	0	0	0	0.1	0.1	0	0	0	0
xls	99.6	95.6	100	99.6	99.9	97.3	98.3	95.3	96.3	99.9
ppt	0	0	0	0	0	0	0	0	0	0
jpg	0.3	4.4	0	0.2	0	0.9	1.6	4.5	2.7	0.1
ogg	0	0	0	0	0	0.2	0	0.1	0	0
png	0	0	0	0	0	0.4	0		0.5	0
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 5: BFA - Fingerprints Trained in 25-50% and tested in 25-50%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	147,705	204,662	285	102,831	201,859	41,013	178,816	193,103	195,368	187,688
pdf	6.9	4	4.9	5.5	5.1	0.1	6	5.8	5.2	5.3
zip	25.2	26.7	14	23.7	28.4	0.6	27.6	30	25.3	29.5
text	32.6	40.1	14.7	30.9	38.8	0.8	33.4	34.7	36.5	37.3
doc	16.3	17.7	4.9	15.6	15.8	0.4	14.8	14.4	19.4	15.1
mp4	2	0.8	2.1	1.1	1.7	0	1.2	1.1	1.1	1.1
xls	3.9	1.7	49.1	11.5	0	97.9	4.2	0.8	1.3	0.4
ppt	9.2	6.7	7.7	8.8	7.4	0.2	9.5	9.8	8.1	8.6
jpg	0.7	0.3	0.7	0.5	0.2	0	0.6	0.6	0.4	0.4
ogg	2.6	1.5	1.8	2	2.2	0.1	2.2	2.2	2.2	2
png	0.6	0.3	0	0.5	0.4	0	0.6	0.5	0.5	0.5
Unclassified	0	0	0	0	0	0	0	0	0	0

ation classified 366,969 fragments as text which corresponds to the 18.9% of the initial corpus. 87,837 of them are fragments that come from pdf, xls, doc and text files and 279,132 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 31.5% of true positives in identifying document-type fragments as text with 68.5% false positives. This percentage of true positives corresponds to the 11.7% of the total pdf, xls, doc and text files of our corpus.

The bad results are probably due to the fact that some of the fingerprints were trained with a tiny amount of fragments, so there are not representative at all, for the category they were build

Table 6: BFA - Fingerprints Trained in 50-75% and tested in 50-75%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	12,421	43	1,203	2,101	15	3,158	2,393	147	66	89
pdf	39.1	23.3	6.2	1.8	0	1.6	1.6	2	3	1.1
zip	4.8	16.3	6.7	10.4	0	0.4	3.1	5.4	1.5	14.6
text	0.6	2.3	1.6	5.9	0	0	2.3	2	0	9
doc	6.2	7	40.9	7.5	0	2.4	18.2	4.1	3	1.1
mp4	12.2	27.9	1.2	37.6	100	27.2	42.6	40.8	36.4	12.4
xls	13.5	0	1.4	19.6	0	65.5	18.7	35.4	15.2	1.1
ppt	16.0	0	17.5	1.4	0	1.5	0.6	0.7	1.5	0
jpg	5.3	0	15.1	1.2	0	1.2	7	1.4	3	3.4
ogg	0.6	0	8.8	3.7	0	0.2	4.9	0	36.4	0
png	1.5	23.3	0.5	10.9	0	0	0.9	8.2	0	57.3
Unclassified	0	0	0	0	0	0	0	0	0	0

Table 7: BFA - Fingerprints Trained in 75-100% and tested in 75-100%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	23,892	0	188,564	20,691	0	1,308	3,053	1,026	0	2
pdf	7.6	0	0.3	0.3	0	0	0.5	0	0	0
zip	0.7	0	0.4	0.5	0	3.7	5.9	1.2	0	0
text	11.8	0	1.4	3.4	0	6.2	2.3	1.8	0	0
doc	2	0	8.2	43.2	0	17.7	5.3	1.4	0	0
mp4	49.3	0	86.5	48.6	0	68.3	78.0	74.4	0	0
xls	7.9	0	0.6	1.2	0	0.9	2.9	0.1	0	0
ppt	0.8	0	0.7	0.1	0	0	0.3	0	0	100
jpg	4.2	0	1.4	1.7	0	1.3	0.8	20.6	0	0
ogg	4.3	0	0.4	0.9	0	1.8	3.7	0.7	0	0
png	11.4	0	0	0	0	0	0.4	0	0	0
Unclassified	0	0	0	0	0	0	0	0	0	0

for. For example it is obvious that in the 0-25% category the xls fingerprint was trained with the 62.83% of the total xls fragments and the ogg fingerprint, for this particular category, was trained with only the 0.02% of the total ogg fragments. Probably this is the reason why in the 0-25% category most of the fragments were classified as xls since most of the other fingerprints, with the only exception of xls, were under-trained. This observation led as to the formulation of the next variation.

11 Variation 3 - Dominant Category Fingerprints

If we look at the table 4.2 it is obvious that most fragments of a certain file type are expected to belong to one of the 4 categories that we discussed in the previous variation. We hypothesized that for every file type the category which contains the majority of files fragments is more representative for the respective file type than the others. So from the 4 fingerprints that we created for every file type for the previous BFA variation, we chose the one which was trained with fragments that belonged in the ratio category with the biggest amount of fragments. We call this category the dominant category of the file type. For example the dominant category of the text file type is the 75-100%, for the pdf is the 25-50% etc. Consequently, we ended up with 10 fingerprints witch corresponds to the dominant categories of every file type. This variation is identical with the first one, with the only difference that we use the fragments of the dominant category of every file type to train our fingerprints instead of the whole fragment set. The results of this BFA variation can be found in Table **??**.

This BFA variation classified 589,402 fragments as text which corresponds to the 30.3% of the initial corpus. 490,267 of them are fragments that come from pdf, xls, doc and text files and 99,135 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 83.2% of true positives in identifying document-type fragments as text with 16.8% false positives. This percentage of true positives corresponds to the 65.3% of the total pdf, xls, doc and text files of our corpus.

Table 8: BFA Results - Dominant Fingerprints

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	189,732	204,795	190,055	177,887	204,728	193,352	195,289	195,608	195,656	195,653
pdf	5.0	3.9	0	2.9	4.9	0	5.3	5.4	4.8	5.1
zip	20.4	26.8	0	13.4	28.2	0.1	25.1	29.5	24.9	28.4
text	27.9	6.8	98.4	51.9	6.4	81.7	17.3	8.6	10.6	9.0
doc	31.4	51.8	0.1	22.1	47.4	0.2	37.5	42.0	47.8	44.6
mp4	3.0	1.9	0	0.9	2.8	0	1.6	1.7	1.4	1.9
xls	1.8	0.3	1.5	2.6	0.4	17.8	1.8	0.4	0.4	0.3
ppt	6.7	6.5	0	4.7	7.2	0	8.5	9.2	7.5	8.1
jpg	1	0.3	0	0.3	0.3	0	0.5	0.6	0.3	0.4
ogg	2.2	1.5	0	1	2.1	0	1.9	2.1	1.9	1.9
png	0.7	0.3	0	0.2	0.4	0	0.5	0.5	0.4	0.4
Unclassified	0	0	0	0	0	0	0	0	0	0

12 Variation 4 - Every fragment with ratio above 75% of our special ASCII subset classified as text

According to the results of table 4.X (Fingerprint Ratio Analysis) almost all text fragments(99.5%) contain more than 75% of our special ASCII subset. In the same ratio category, fragments of pdf, doc and xls correspond to 15.66%, 10.98% and 1.28%, of the total amount of fragments of their particular file type, respectively. All other file types have less than 1% of their total fragments in this ratio category. We thought that it would be interesting to apply the BFA of variation 1 only to the fragments which have less than 75% of our special ASCII subset and every fragment above this percentage would be classified as text. We should note that we decided to use the fingerprints of variation 1 instead of the dominant fingerprints of variation 2, because overall percentage of text fragment classification is better for variation 1. The results of this variation of BFA can be found in Table **??**.

This BFA variation classified 590,834 fragments as text which corresponds to the 30.4% of the initial corpus. 512,855 of them are fragments that come from pdf, xls, doc and text files and 77,979 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 86.8% of true positives in identifying document-type fragments as text with 13.2% false positives. This percentage of true positives corresponds to the 68.3% of the total

Table 9: BFA - Fingerprints Trained in 0-75% and tested in 0-75%

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
num.of fragments	165,840	204,795	1,491	157,196	204,728	192,044	192,236	194,582	195,656	195,651
pdf	31.5	52.3	3.5	22.9	48.1	0.2	35.9	40.9	46.5	44.1
zip	21.6	26.6	2.7	15.0	28.0	0.1	25.2	29.4	24.7	28.2
text	15.2	4.9	26.4	44.1	4.4	95.5	13.1	5.5	7.1	7.2
doc	16.0	4.2	59.6	7.9	5.2	0.2	9.7	7.9	8.7	5.8
mp4	0.6	0.6	0.1	0.3	0.8	0	0.4	0.5	0.4	0.5
xls	1.2	0	5.0	0.8	0.1	3.9	0.8	0.2	0	0.1
ppt	3.5	2.2	1.1	2.1	2.7	0	3.4	3.3	2.7	2.9
jpg	0.1	0.1	0.1	0.1	0	0	0.1	0.1	0	0.1
ogg	2.8	2.2	0.7	1.6	3.0	0	2.8	3.0	2.7	2.7
png	7.5	6.9	0.8	5.2	7.7	0	8.5	9.2	7.2	8.5
Unclassified	0	0	0	0	0	0	0	0	0	0

pdf, xls, doc and text files of our corpus.

13 Optimal Variation for Text Fragment Classification

It is obvious that the second variation is by far the worst and cannot aid the design process of our classification algorithm. Among the other three variation, variation 4 yields the best results. Both coverage and accuracy of variation 4 is undoubtedly the highest among the other two. However, taking under account that these are results from a controlled corpus and not from a real life scenario, the fact that variation 4 classifies every fragment with more than 75% ratio of our special ASCII subset as text is a major weakness.

In a real life scenario, the ratio between the amount of fragments of every file type it is highly unlikely to be 1:1, as it is in our corpus. Therefore in a scenario where the corpus does not contain any text fragments, every fragment with a ratio higher than 75% of our special ASCII subset will be falsely classified as text. Furthermore, our corpus is comprised only of 10 file types. Considering the fact that the number of file types that a forensic practitioner is likely to encounter in real life cases is way bigger, renders variation 4 unscalable. We should conduct similar research for all file types first, in order to be able to say if variation 4 can be used in actual forensic cases. Among the remaining variations, variation 1 is slightly better in both coverage and accuracy than variation 3. We judge that this is the optimal variation of BFA for text fragment classification and will be used as the initial phase of our classification algorithm.

14 BFA Training - complete ASCII set VS plain text

Although BFA variation 1 yielded the best results regarding text fragment classification among the other 3 variation, a comparison with a BFA which use the complete ASCII byte set is essential, in order to choose which approach is the best for the design of our algorithm. Ashim[] tested a BFA for fragment classification using the exact same file types as we did. The only exception is that he used the whole ASCII byte set for the fingerprint training. The corpus that he used is almost 10 times bigger than the one we used for training. Conveniently enough, he trained his fingerprints with 10%, 20%, 50% and 100% of his training data set and provided the accuracy results. Our training set, around 800mb for each file type, is approximately the 10% of Ashims training set. In order to have a more objective comparison, we are going to compare the results that Ashim got by using fingerprints which were trained with the 10% of his training set, with our BFA variation 1. This way fingerprints from both approaches received the same amount of training. The results can be found in Table **??**.

For broad fragment classification, fingerprints that use the whole byte set seems to be way more effective than variation 1. Only the accuracies for pdf and ppt are higher in variation 1, simply because Ashims BFA achieved 0% of true positives for these file types. Regarding text fragment classification the accuracy results are pretty close. We took the accuracy percentages that correspond to text fragment classification from Table **??** and calculated the amount of fragments that would be classified as text using this technique. We should mention that since Ashims BFA is not limited to classify fragments that do not contain plain text, the amount of fragments that his BFA could process is bigger(Table X). According to this, that BFA would have classified 462,345 fragments as text which corresponds to the 22.3% of the initial corpus. 410,173 of them are fragments that come from pdf, xls, doc and text files and 52,172 fragments originate from the other 6 file types. This means that in the set that is classified as text we have an 88.7%

Table 10: BFA Results - Training with complete ASCII byte set

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
pdf	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
zip	33.6	86.0	1.9	17.9	22.0	0.0	48.1	33.5	6.7	62.8
text	15.7	0.1	96.2	47.7	4.7	43	5.5	1.1	10.4	2.3
doc	2.1	0	0	0.5	0.6	0	0.4	0.1	8.2	0.3
mp4	10.1	4.5	0.4	4.1	27.2	0	12.3	25.2	18.2	11.4
xls	11.4	0.3	0.3	17.9	0.2	56.8	10.9	4.4	6.4	1.8
ppt	0	0	0	0	0	0	0	0	0	0
jpg	2.6	1.3	0.2	2	0.2	0	4.6	9.7	3.4	1.9
ogg	20.6	3	0.2	6.5	39.7	0	10.9	16.3	40.2	6.4
png	4.1	4.5	0.4	2.8	5	0	6.8	9.4	6.2	12.8
Unclassified	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

of true positives in identifying document-type fragments as text, with 11.3% false positives. This percentage of true positives corresponds to the 50.3% of the total pdf, xls, doc and text files of our testing set(fragments with no plain text included).

Although the accuracy of Ashims results is slightly higher(88.7%) from variation 1(85%), the amount of document-type fragments that is classified as text is significantly lower. Variation 1 classified as text 501,012 of the total pdf, xls, doc and text fragments, in comparison to Ashims BFA that would have classified 410,173. By using BFA as the first phase of our algorithm, we aim to retrieve us much pdf, xls, doc and text fragments as possible and maximize. In that case, this is a trade-off between accuracy and the amount of document-type fragment retrieval. Accuracy levels are pretty close. However, variation 1 classifies significantly more(22%) pdf, xls, doc and text fragments of the total corpus as text. For that reason, we chose to use variation 1 over a BFA which use trained fingerprints with the complete ASCII byte set, for the initial design phase of our classification algorithm.

Classification Metrics

15 BFA Variation 1 Output

After the run of variation 1 BFA, we isolated all fragments which were classified as text. Initially, we expected that BFA falsely classifies fragments from non-text files as text, due to their high plain text concentration. We conducted a plain text concentration analysis on the BFAs output and it seems that BFA classified as text fragments with diverse plain text concentration. This analysis can be found in Table **??**.

Although the 85% of BFAs output originates from document-type files, our algorithms considers all these fragments to be of xls, pdf, doc and text type. By doing this, we expect that the amount of fragments that were falsely classified as text without belonging to a document-type file, will be evenly distributed among the false positive classification results for xls, pdf, doc and text fragments. Our algorithms goal is to be able to correctly identify and distinguish between xls, pdf, doc and text fragments. For that purpose we conducted statistical analysis in the BFAs output trying to find patterns that will help as increase our algorithms accuracy. We introduce two new metrics, the individual null byte frequency and the plain text ratio category. The individual null byte frequency in conjunction with shannon entropy[] can be used to effectively distinguish between pdf from xls and doc fragments. Additionally, the plain text ratio category metric can be used to prevent our algorithm to falsely classify a fragment that belongs to a cer-

Table 11: BFAs Output Plain Text Concentration Analysis

ratio	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
0 - 25%	5,606	75	3	51,462	2,606	145,106	9,920	1,178	198	7,781
25 - 50%	14,008	9,901	127	16,474	6,395	35,315	13,867	9,328	13,715	6,352
50 - 75%	5,646	11	263	1,416	0	2,974	1,396	110	53	13
75 - 100%	15,115	0	185,952	20,247	0	1,298	2,373	1,025	0	0
Total:	40,375	9,987	186,345	89,599	9,001	184,693	27,556	11,641	13,966	14,146
0 - 25%	13.9	0.8	0	57.4	29	78.6	36	10.1	1.4	55
25 - 50%	34.7	99.1	0.1	18.4	71	19.1	50.3	80.1	98.2	44.9
50 - 75%	14	0.1	0.1	1.6	0	1.6	5.1	0.9	0.4	0.1
75 - 100%	37.4	0	99.8	22.6	0	0.7	8.6	8.8	0	0

tain plain text ratio category.

16 Plain Text Concentration Categories

As we already mentioned, file fragments of certain types are expected to have certain plain text concentration. We use 4 concentration categories of equal size. 0-25%, 25-50%, 50-75% and 75-100%. Our metric assumes that fragments are of 512-bytes size. As we saw in Table **4.2**, 75% or more of text fragments is plain text and the majority of xls fragments(97%) are 0-50% plain text. Moreover more than 90% of the total mp4, zip, ppt, jpg, png and ogg fragments are 25-50% plain text. Additionally, we run an extra analysis specifically for the text fragments and we found that 98% of them are fully comprised of plain text. We are positive that this light weight metric can be combined with current techniques and increase their accuracy. For example if a fragment is classified as text and it contains at least one non-plain text byte, then probably it's not a text fragment. So a classification algorithm could make this simple check and substitute its first classification "guess" with the one that had the second highest accuracy level. Similarly, if a fragment is more than 75% plain text then probably it's not a mp4, zip or ogg fragment etc.

17 Individual Null Byte Frequency

We applied several statistical metrics such as median, mean, mode, standard deviation, minimum and maximum frequency byte values in the BFAs output fragments. However, we couldn't find something extra that could significantly aid our algorithms design. Thereafter, we manually inspected several fragments from all the file types, and we noticed that the amount of null bytes in xls fragments was significantly high. However, although slightly less, the frequencies of null bytes was also similar for doc and pdf fragments. We noticed that there were many long sequences of null bytes in most of the pdf and doc fragments but in the xls fragments these sequences were significantly fewer. Additionally, the majority of the total null bytes in xls fragments were individual. Therefore, we analysed the distribution of individual null bytes for all the document-type fragments. As you can see in figures 1,2,3,4 the number of individual null bytes in xls fragments is obviously higher than the one of the other file types. For text fragments, the amount of individual null bytes is 0 and for pdf and doc fragments the frequency mainly ranges from 0 to 25. Since the majority of text fragments are fully comprised of plain text, it's natural that they do not contain null values.

18 Shannon Entropy

There is a widespread use of the Shannon entropy[Shannon] metric in file carving techniques. Entropy measures how much information a sequence of symbols contains[Shannon,Calhoun]. Entropy is defined as:

$$H(X_i..X_n) = -\sum_{i=0}^{n} p(x_i)\log_2 p(x_i)$$

In our case, $X = X_i..X_n$ is the byte-content of a fragment, where $n = 511$ and $p(x_i)$ is the frequency of x_i in X. To calculate $p(x_i)$, we simply divide the number of occurrences of x_i in a fragment with the fragments size. It is known that usually compressed files have high entropy in contrast with text files that have low entropy[Calhoun,Jeroen-Thesis]. Since pdf is a compressed file format, we expected that pdf fragments will have significantly higher entropy than doc, xls and text fragments. In figures 1,2,3,4 we can see the entropy distribution among these file fragments. Most of the pdf fragments have an entropy value of 6 or more, in contrast with the other file-type where the majority of their fragments has an entropy of value 6 or less.

Algorithm Description

Results

Longest Common Subsequence While trying to find a way to reduce false positives of the doc and xls fragment classification, we thought to test the performance and accuracy of the longest common subsequence technique. Calhoun[] used this technique to distinguish between fragments of two different file types. He achieved high accuracy results(90%) using the standard dynamic programming version of the algorithm. Even though the dynamic version is faster than the naive approach of the algorithm, with runtime complexity mxn, where mn the length of the input strings, it still seems like an "expensive" technique to be used in file carving. He extracted the longest common subsequences of every file fragment in his training set and concatenated them in a big string. This string is used as a representative of the respective file type. Due to the fact that the speed of this technique depends on the length of the input strings, it is essential to know about how long the file type representative string should be in order to be effective. Since he does not provide information about the length of the strings that he used as file type representatives , we want to find out strings of what length can be used as file type representatives and yield similar results. If the lengths are not too long then the computation of the longest common subsequence between two strings could be fast enough to be used in file carving techniques.

Instead of concatenating every longest common subsequence between fragments of the same file type, we tried a different approach. We used 500 fragments of the doc and xls type for our representative string creation. This resulted to $500x500 - 500 = 249,500$ comparisons for each file type. We gathered all longest common subsequences from these comparisons and putted them in a map data structure. Then we sorted the map and took the first 100, 500, 1000 and 1500 most frequent longest common subsequences. We concatenated these subsequences in 4 long representative strings for each of the doc and xls file type. Thereafter, we used a set of 10,000 fragments, 5000 of xls and 5000 of doc type, to test their accuracy. At this point we should note that Calhoun used only 50 fragments per file type to test this technique. This was an additional reason to want to try its performance since we consider testing sets of this size extremely insufficient. However, we also consider our testing set significantly small, but since our goal was to test the correlation between the speed and the accuracy of that technique, that size is acceptable. The results can be found in Table **??**.

Table 12: Longest Common Subsequence comparison for doc vs. xls

n most frequent lcs	$n = 100$	$n = 500$	$n = 1000$	$n = 1500$
doc vs. xls precision	83	89.5	90.06	91.63
doc lcs representative string length	1,007	5,763	15,225	27,070
xls lcs representative string length	859	4,679	9,482	14,609

As someone would expect, using longer strings as file type representatives result in higher classification precision. The precision gradually increases while using longer strings. However, Although the precision of this metric proved to be in pair with the results Calhoun presented [], its speed is way to slow to be used in real life cases. Even by using the shortest file type representative strings, which corresponds to the first 100 most frequent longest common subsequences of a file type, the runtime complexity remains extremely high. We compared the speed of this technique with our unoptimized algorithms speed, and although our benchmarking is not completely accurate, the longest common subsequent technique takes 56% more time to compute than our complete algorithm(BFA included). Taking under account that our algorithm was designed to be able to handle 10 different file types and that the LCS technique that we tried only 2, it is obvious that the difference in speed is quite significant. Moreover, since our algorithm yielded few false positives for xls fragment classification, we don't think that we could improve our overall accuracy by using the LCS technique. In conclusion, we strongly believe that such an expensive technique is not appropriate for broad fragment classification and researchers should first invest time in searching for light weight techniques before trying brute force approaches.

Conclusion

Acronyms

FTA Fault tree analysis

MTTF Mean time to failure

RAMS Reliability, availability, maintainability, and safety

Additional Information This is an example of an Appendix. You can write an Appendix in the same way as a chapter, with sections, subsections, and so on.

A Introduction

A.1 More Details