University of Amsterdam

Master Thesis

A Specialized Approach for Document-Type Fragment Classification in Digital Forensics

Author:
Ioannis Tzanellis

Supervisors:
Jeroen van den Bos
Tijs van der Storm

A thesis submitted in fulfilment of the requirements for the degree of MSc in Software Engineering

July 25, 2013

This page intentionally left blank.

Acknowledgements

I would like to express my gratitude to my supervisor Jeroen van den Bos for the initial subject of this thesis and his constant support and engagement. Our cooperation was excellent.

Furthermore, I would like to thank all the academic staff of the Software Engineering Master Programme who were always eager to listen and to provide their insight. Also, I like to thank all of my colleagues in CWI from the SWAT group and in particular Ashim Shahi and Davy Landman who were always available to help me with everyday little issues. Last but not least, I would like to thank my family. Without their support this Master thesis would not have been possible.

Contents

Acknowledgements i

Tables iv

Figures v

1 Introduction 1
1.1 Background . . . 1
1.2 Problem Formulation . . . 2
1.3 File Fragment Classification . . . 3
1.4 Objectives . . . 3
1.5 Algorithm Requirements . . . 4
1.6 Methodology . . . 4

2 Related Work 6

3 Experimental Setup 8
3.1 Byte Frequency Analysis(BFA) Algorithm . . . 8
3.2 Data Set . . . 9

4 Algorithm Development 11
4.1 Approach Description . . . 11
4.2 BFA Variations . . . 12
4.2.1 Variation 1 - Plain Text ASCII Subset . . . 12
4.2.2 Variation 2 - Plain Text Concentration Categories . . . 12
4.2.3 Variation 3 - Dominant Plain Text Concentration Categories . . . 14
4.2.4 Variation 4 - Fragments above 75% Plain Text Concentration classified as text . . . 15
4.2.5 Optimal Variation for Text Fragment Classification . . . 16
4.2.6 BFA Training - Entire ASCII Set VS. Plain Text . . . 16
4.3 BFA Text Output Analysis . . . 18
4.3.1 BFA Variation 1 Output . . . 18
4.3.2 Plain Text Concentration Categories . . . 19
4.3.3 Shannon Entropy . . . 19
4.3.4 Individual Null Byte Frequency . . . 20
4.4 Longest Common Subsequence . . . 22

5 Algorithm Description 24

6 Results **29**
6.1 BFA Scan - Text Fragment Classification 30
6.2 BFA Extension Algorithm Accuracy 30
6.3 Complete Algorithm Accuracy 31

7 Analysis **32**
7.1 BFA performance 32
7.2 Overall Algorithm Performance 33
7.3 Algorithm Comparison 34
7.3.1 Accuracy Comparison 35
7.3.2 Execution Time Comparison 35

8 Discussion **37**

9 Conclusions and Future Work **39**
9.1 Conclusions 39
9.2 Future Work 40

A BFA Variation 2 - Results **42**

B BFA Output - Histogram Analysis **44**

Bibliography **47**

Bibliography **47**

Tables

3.1 Experimental Data Set 9
3.2 Final Testing Data Set 10

4.1 BFA Results - Fingerprints trained with plain text ASCII subset 12
4.2 Training Set - Plain Text Concentration Analysis 13
4.3 BFA Results - Dominant Fingerprints 14
4.4 BFA - Fingerprints Trained in 0-75% and tested in 0-75% 15
4.5 BFA Variation Comparison 16
4.6 BFA Results - Entire ASCII Byte Set Training 17
4.7 BFA Output Plain Text Concentration Analysis 18
4.8 Longest Common Subsequence comparison - doc VS. xls 23

6.1 BFA Text Fragments Classification 30
6.2 BFA Extension Algorithm Accuracy 30
6.3 Algorithm Accuracy Results 31

7.1 Algorithm Prediction Rates 34
7.2 Classification Algorithm Accuracy Comparison 35
7.3 Classification Algorithm Average Execution Time 36

A.1 BFA - Fingerprints Trained in 0-25% and tested in 0-25% 42
A.2 BFA - Fingerprints Trained in 25-50% and tested in 25-50% 42
A.3 BFA - Fingerprints Trained in 50-75% and tested in 50-75% 43
A.4 BFA - Fingerprints Trained in 75-100% and tested in 75-100% 43

Figures

1.1 Hard Drive Fragment Representation 3

4.1 Entropy Distribution 20
4.2 Individual Null Byte Distribution 21

5.1 Algorithm as Decision Tree 28

6.1 Algorithm Overview 29

B.1 Individual Null Byte Distribution 45
B.2 Entropy Distribution 46

1 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 metadata in order to track the location of their files. Metadata can contain information such as creation date, data structure (e.g directory or regular file), file type, file owner, size, last modified date and more. In practice, most data can be recovered using the regular file system, but often investigators are specifically interested in the data that appears to be missing. In a real life forensic case, it is likely that a part of file system metadata might be corrupted or deleted.

1.1 Background

File carving is a forensic technique that recovers files based on their content, without relying on their metadata. The file carving process involves two steps. File format validation and file reconstruction [1]. During the recovery procedure, forensic investigators must first validate the type of the file and then apply the appropriate reconstruction technique. At this point we should note that in this thesis, only the file format validation techniques are of our interest. By examining the complete byte content and/or some characteristic byte sequencies of an unclassified file, file format validation techniques are used to classify its type. The Magic Number Matching technique [16] looks for magic numbers, specific byte sequencies that signal the beginning and/or the end of a file (headers,footers) and try to classify them to a file type according to that information. For example jpeg files begin with the hexadecimal sequence 0xFFD8 and end with 0xFFD9 [15]. Similarly, the Data Dependency Resolving technique is used to identify fields that specify file attributes like color or size [1].

Furthermore, other file carving techniques use statistical learning algorithms, which process the complete byte set of a file, creating a representative fingerprint for every file

type. A classifier compares these fingerprints with an unidentified byte sequence and produce an accuracy level for each fingerprint. Then, it classifies the unidentified byte sequence to to the file type of the fingerprint that yielded the highest accuracy level. Some common statistical learning techniques are the n-Gram Analysis [12] and the Byte Frequency Analysis (BFA) and the Byte frequency cross-correlation(BFC) algorithms [14].

1.2 | Problem Formulation

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 characteristic structures [14]. 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 overall internal structure and characteristics.

So why is this a problem? The answer lies in file systems behaviour and file fragmentation. When we delete a file from a data storage device, 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 [15]. That means the next time a new file is created, the file system is free to use these unallocated sectors to store a new one. 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 won't have enough space to store it. So the file system will allocate-overwrite all the sectors of the previous deleted file, while the remaining data which do not fit, will be stored in other unallocated sectors. This results to file fragmentation.

Although fragmentation in current file systems is small [9], the classification of the missing fragmented parts of a file are essential for its recovery. In that case, validation techniques which use the complete file content are unable to provide aid to forensic examiners.

1.3 File Fragment Classification

File fragment classification is a technique that uses only a small fragment of a file, in order to determine its type. This approach is independent from files overall structure. In Table 1.1 we can see a hard drive representation as a sequence of byte-blocks. Each block in our figure is a file fragment of a certain file format that is defined by a capital letter.

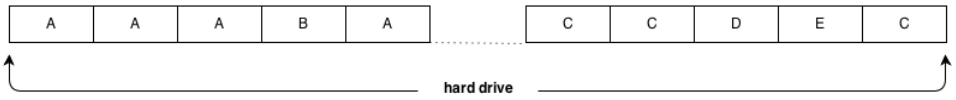

FIGURE 1.1: Hard Drive Fragment Representation

Although in theory, file fragment classification looks like an ideal approach, in practice it proved to be difficult to create a technique of high precision [19]. In the last Digital Forensic Research Workshop (DFRWS 2012) challenge, the winning classification tool achieved an overall classification accuracy of 36% [13], in a corpus of 38 different file types.

1.4 Objectives

The main objectives in this project are:

1. Create a classification algorithm for identifying document-type fragments of higher precision than the existing similar algorithms. In particular, we focus on the classification of text, xls, doc and pdf file fragments and try to improve their classification rates.

2. Test the hypothesis that by analysing only a special ASCII byte-set of file fragments which corresponds to the printable ASCII characters, accuracy of classification algorithms can be enhanced for document-type fragments. This ASCII subset is comprised of byte values of the range $32 \leq b \leq 126$ along with the tab (9), new line (10) and carriage return (13) bytes. From this point, we will refer to this special ASCII subset as "plain text".

1.5 | Algorithm Requirements

The design requirements of our classification algorithm are as follows:

1. Speed - Comparable in runtime performance to the current light-weight algorithms such as the N-Gram Analysis [12] [19] and the BFA algorithm [14] [19].
2. Accuracy - Improve upon the overall accuracy of the algorithms in the same runtime performance class.
3. Operate in common fragment sizes, preferably of 512-bytes size, the smallest relevant size equivalent to a hard drives sector size.

1.6 | 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 that 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 plain text ASCII subset of a fragment, we could achieve better results regarding text fragment classification. The plain text characters are a behavioural trait of a document, so we expect that their analysis might reveal their file type.

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 significant complexity and to create a custom more effective version of it. Our algorithm of choice is the Byte Frequency Analysis [14]. The main reason of that choice is that it's been observed that BFA falsely classifies a big amount of document-type fragments as text. We will elaborate more about the reasons of this choice in Chapter 3.

Our design procedure is comprised of two main phases. In phase 1 we intent to use a BFA that analyses only the plain text byte set for a fast scan of the corpus, in order to isolate a big amount of document-type fragments. In phase 2 we analyse the complete ASCII byte set of BFA output and try to classify the file type of the document-type fragments.

During phase 1, we use 4 variations of BFA that analyse only the plain text content of the input fragments and test our hypothesis. We compare the results of these variations with each other and we choose the one that yields the best results regarding document-type fragment classification. By document-type classification, we mean the classification of fragments of the pdf, text, doc and xls file formats. After that we compare the best BFA variation with the BFA variation that Shahi used in [19] for fragment classification, which takes under account the entire ASCII byte set. Finally, we choose the variations that yields the best results. This BFA variation will be the first part of our final classification algorithm. Thereafter, we isolate all fragments that were classified as text, resulting in a new data set and proceed in phase 2 of our design procedure.

During phase 2, we analyse the whole byte content of fragments that were classified as text, trying to find patterns that could aid our algorithms design. Initially we used simple statistical metrics such as the mean, mode, median and standard deviation trying to find characteristic patterns in specific file types. This resulted to focus on some specific byte values, where in conjunction with the histogram analysis we did, resulted in the formulation of two new metrics. The Individual Null Byte Frequency (INBF) and the Plain Text Concentration (PTC). We combine these two metrics along with the Shannon entropy metric [20] and the accuracy levels that BFA produced in phase 1, to create a new custom algorithm.

2 Related Work

Karresand and Shahmehri [11] introduced a new algorithm that uses the Rate-of-Change (ROC) metric. They define the rate of change of a byte sequence as the difference of the ASCII values of consecutive bytes. Although this technique yields good classification rates for jpeg files (99% true positives), mainly because of their 0xFF00 metadata tags, for other files types the false positive rates are extremely high (e.g for zip and portable executable (PE) files near 70% false positives rates).

Veenman [22] used a combination of BFA [14] with Shannon entropy and Kolmogorov complexity measures to classify fragments that were 4096 bytes in size. He used a corpus of 450MB comprised of 11 different file types. He managed to achieve high detection rates(99%) for jpeg and html files. However, results for the other file types weren't so good, achieving an overall accuracy of 45%. Additionally, the big size of the fragments that Veenman used is not convenient enough for a real forensic case.

Calhoun and Coles [4] used a set of techniques like byte frequency of ASCII codes and Shannon entropy, linear discriminant analysis and prediction with longest common substrings and subsequences along with many other common statistical metrics. Their corpus was comprised of gif, pdf, jpeg and bmp files. Although they achieved high precision rates of 88.3%, their testing set was significantly small, comprised only of 50 fragments per file type. The fragments size that were used in their experiment was of 512 and 896 bytes. Moreover, since they don't give information about the lengths of the file type representative strings that were used, we don't know how expensive the longest common subsequence technique can be.

Axelsson [2] used a corpus of 28 different file types and applied the k-nearest-neighbour classification technique with Nearest Compression Distance (NCD) as the distance metric between file fragments. The results are unremarkable, achieving an average accuracy of around 34%. It was observed that this approach achieved higher accuracy for fragments with high entropy.

Li et al. [12] used the N-Gram Analysis to create representative fileprints for file types. The fileprints were based on a centroid which combined the mean and the standard deviation of byte frequencies. More specifically, they focused on 1-Gram Analysis of the ASCII byte values, representing a file as a 256-element histogram. In order to compare an unknown byte stream with a fileprint they used the Mahalanobis distance function. When they applied this technique in full files they achieved true positive rates of 60-90%. Moreover, by using only the first 20 bytes of files they managed to achieve an accuracy of 99%. This was due to the fact that these 20 bytes mainly contained header data (Magic Numbers). By using the same approach in entire files the accuracy dropped significantly. It is quite paradoxical that by using more data they managed to get less accurate results.

Fitzgerald, Mathews, Morris and Zhulyn [7] investigated whether techniques from natural language processing could be applied successfully to file fragment classification. They used the macro-averaged F1 metric in a set of 24 file types. They achieved an average prediction accuracy of 49.1% on 24 file types outperforming Axelssons (34% for 28 file types) and Veenmans (45% for 11 file types) results.

Roussev and Garfinkel [18] claimed that specialized approaches must be used in file fragment classification in order to produce practical tools. They studied how some popular file formats are structured by the use of several case studies and reached to the conclusion that existing generic file fragment classification methods are unlikely to be successful. Furthermore, they suggest that by manually analysing files and studying their standards, effective custom classifiers can be created. Inspired by this conclusion, this thesis describes a file fragment classification approach that doesn't attempt to classify all file formats using a single algorithm. However, we do attempt to find a single approach to classify multiple formats.

Lastly, Shahi [19] tested 4 different classification algorithms in the same corpus, in order to compare their performance. His corpus was comprised of 10 different file types. The algorithms used were the BFA [14], the N-Gram Analysis [12], the Rate of Change [11] and the algorithm of Conti et al. [5]. The results show that the average overall accuracy of the aforementioned techniques is around 30%. Moreover, Shahi benchmarked their performance in terms of execution time and found out that the N-Gram Analysis is the fastest among them, with BFA coming second, Rate of Change third and fourth the algorithm of Conti et al.

3 Experimental Setup

3.1 Byte Frequency Analysis(BFA) Algorithm

BFA [14] is a statistical learning algorithm that was initially developed to analyse and classify whole files. It was not meant to be used for file fragment classification. By counting the number of instances of each byte in a file of a certain type, BFA creates a representative average value for each byte instance, along with its respective correlation strength. This results in a fingerprint for a particular file-type. Thereafter, during classification procedure, the input file is compared with every fingerprint and an accuracy level is created for each of them. BFA classifies the file to the file type of the fingerpint that corresponds to the highest accuracy level.

Shahi [19] trained and tested a BFA with file fragments of 512-byte size. His results show that although the algorithm is pretty bad for broad fragment classification, it is quite good in classifying fragments of document-type files, as text. Additionally, he tested the performance of BFA along with the Rate of Change, n-Gram Analysis and Conti et al. algorithm. The results show that BFA has the highest precision in classifying document-type fragments as text.

In contrast to the default technique, we use a BFA that trains the fingerprints with byte values that correspond only to the plain text ASCII characters, instead of the complete byte-set of the fragments. Moreover, we also use fragments of 512-bytes size. This BFA will be the first half of our final classification algorithm and after this point we intend to use additional metrics to create a custom classifier. Taking under account speed requirements, BFA seems as a good candidate since it is a light-weight technique, compared to similar statistical learning algorithms [19] or heavier machine learning techniques.

3.2 Data Set

The data set we use for our training, experimentation, analysis and testing procedures is derived from Garfinkels [9][24] corpus, Wikipedia and Academic Earth [23] and is a subset of the coprus that Shahi used in [19]. Our corpus is comprised of 10 different file types with a total size of about 20GB. We divided the corpus in half, resulting in two subsets of 10GB each. The experimental and the final testing set.

We use the experimental set to do all of our experimentations, analysis and training and the final testing set for testing the performance of our final algorithm. At this point we should note that the 10GB that corresponds to the final testing set won't undergone any type of analysis that will affect the design of our algorithm, since we only want to use it for testing its performance. We fully designed our algorithm based on the experimental 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 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 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
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 fragments	204,800	214,835	199,475	205,209	214,835	205,209	206,028	205,209	205,209	207,872
Output fragments	189,732	204,795	190,055	177,887	204,728	193,352	195,289	195,608	195,656	195,653
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

TABLE 3.1: Experimental Data Set

In the experimental set, we split these 10GB in two subsets of 9-1 ratio. 90% of the experimental set is used as our training set and the other 10% as our experimental testing set. Additionally, we transformed all of our file contents, in both the experimental and the final testing set, into 512-byte blocks, which we refer to them as fragments. Since our algorithm would be able to classify only fragments that contain at least one plain text character, fragments with no plain text were discarded. The percentage of discarded fragments per file type can be found in Table 3.1. As we can see, the percentage of fragments with no plain text for most of the file types is around 5%. I is quite interesting that this percentage is significantly higher (10-17%) for the doc file type.

Furthermore, we use our training set to train our fingerprints and the experimental testing set to test the 4 BFA variations. Both of the aforementioned sets undergone statistical analysis in order to discover useful patterns. More detailed information about our experimental data set can be found in Table 3.1. Moreoever, information regarding the final testing set that we use to test the performance of our final algorithm can be found in Table 3.2.

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
Size in megabytes	966.2	967.4	968.8	968.4	969.5	968.6	968.1	967.4	969.4	969,6
Num.of files	1,800	19	1,496	1,865	18	624	816	3,352	1,833	4,948
Expected fragments	1,978,777	1,981,235	1,984,102	1,983,283	1,985,536	1,983,692	1,982,668	1,981,235	1,985,331	1,985,740
Output fragments	1,874,910	1,889,477	1,891,472	1,775,747	1,888,605	1,870,376	1,864,145	1,886,853	1,891,754	1,880,742
Fragments with no plain text(%)	5.2	4.6	4.7	10.5	4.9	5.7	6	4.8	4.7	5.3

TABLE 3.2: Final Testing Data Set

4 | Algorithm Development

4.1 | Approach Description

Our algorithms development procedure is comprised of two main phases. In the first phase, we use 4 different variations of BFA using only the plain text ASCII byte set of the fragments. We compare these 4 BFA variations and we choose the one that yields the best results regarding text fragment classification. Additionally, we compare the performance of our best BFA variation with Shahis [19] BFA variation. In the end of phase 1 we choose the best technique regarding text fragment classification and proceed to phase 2.

In phase 2, we isolate all fragments that were classified as text from the optimal BFA variation, in order to analyse them. This analysis resulted in the formulation of 2 new light-weight classification metrics, where in conjunction with the Shannon entropy [20] metric and the file type accuracy levels of BFA, aid the design of our classification algorithm.

Finally, in subsection 4.4 we test the Longest Common Subsequence technique in order to find out what is the runtime cost that we have to "pay" in order to aquire similar results as Calhoun did in [4] and decide if it could aid our algorithms design.

4.2 BFA Variations

4.2.1 Variation 1 - Plain Text ASCII Subset

In this variation we created 10 fingerprints, one for each file type, which were trained with fragments from the training set. We only the printable ASCII characters byte-set for the fingerprint training. The results can be found in Table 4.1.

This BFA variation classifies 589,758 fragments as text which corresponds to the 30.4% of the initial corpus. 501,012 of them are fragments that originate 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 positive rates in identifying document-type fragments as text with 15% false positives. This 85% of true positive rates corresponds to the 66.7% of the total pdf, xls, doc and text files of our corpus.

	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	20.3	48.1	0.2	35.3	40.7	46.5	44.1
zip	20.2	26.6	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	0.2	0.8	0	0.4	0.5	0.4	0.5
xls	1.2	0	1.4	0.8	0.1	3.9	1.0	0.2	0	0.1
ppt	3.2	2.2	0	1.8	2.7	0	3.3	3.3	2.7	2.9
jpg	0.5	0.1	0	0.1	0.0	0	0.1	0.1	0	0.1
ogg	2.8	2.2	0	1.4	3.0	0	2.8	3.0	2.7	2.7
png	6.8	6.9	0	4.6	7.7	0	8.3	9.1	7.2	8.5
Unclassified	0	0	0	0	0	0	0	0	0	0

TABLE 4.1: BFA Results - Fingerprints trained with plain text ASCII subset

4.2.2 Variation 2 - Plain Text Concentration Categories

During our research we thought that it would be interesting to analyse the distribution of the plain text ASCII byte values. Depending on its plain text concentration, a fragment is assigned to one of 4 plain text concentration categories. 0-25%, 25-50%, 50-75% and 75-100%. The results of this analysis can be found in Table 4.2. We should note that fragments which do not contain plain text are excluded from this analysis.

As it seems, fragments from certain file types are more likely to belong to certain concentration categories. For example, almost all text fragments (99.95%) contain more than 75% of plain text byte values and almost all xls fragments less than 50%. Undoubtedly

this is completely reasonable. Text files are mostly comprised of plain text while Excel sheets, due to the space that their cell structure occupies, contain less printable characters. That finding can be used as a metric to improve current classification techniques. We will further elaborate on this later in this chapter.

Based on the analysis results, we thought that it would be interesting to divide the fragments of our training set in 4 plain text concentration categories. Then for each category and for each file type we created their respective fingerprints. So we ended up with 40 fingerprints, 4 for each file type. The algorithm first checks the plain text concentration of the input fragment and according to its value, it compares the fragment with the corresponding fingerprint. For readability purposes, we placed the result tables in Appendix A.

The accuracy for both the actual classification and the text classification are really bad. This variation classified 366,969 fragments as text which corresponds to the 18.9% of the initial corpus. 87,837 of them are fragments that originate 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 positive rates in identifying document-type fragments as text with 68.5% false positives. This percentage of true positive rates 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. Therefore, they are not representative at all for the category they were build 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 only with 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.

plain text concentration (%)	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
$0 < c \leq 25$	0.55	0.02	0.01	36.03	0.20	62.61	6.83	0.46	0.05	0.70
$25 < c \leq 50$	78.68	99.96	0.03	52.39	99.80	34.24	91.93	99.20	99.94	98.91
$50 < c \leq 75$	5.11	0.02	0.01	0.60	0	1.88	0.60	0.08	0.02	0.11
$75 < c \leq 100$	15.66	0	99.95	10.98	0	1.28	0.64	0.26	0	0.29

TABLE 4.2: Training Set - Plain Text Concentration Analysis

4.2.3 Variation 3 - Dominant Plain Text Concentration Categories

If we look at Table 4.2 it is obvious that most fragments of a certain file type are expected to belong to one of the 4 plain text concentration categories that we discussed in the previous variation. We hypothesized that for every file type the category in which the majority of of fragments of this particular file type belong is more representative for the respective file type than the other categories. So from the 4 fingerprints that we created for each of the 4 plain text concentration categories, we chose the one that was trained with the biggest amount of fragments for its particular file type. We call this category the dominant plain text concentration category of the file type. For example the dominant plain text category of the text file type is the 75-100% category, for the pdf is the 25-50%, for the xls is the 0-25% etc.

Consequently, we ended up with 10 fingerprints witch correspond to the dominant plain text 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 categories 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 4.3.

	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

TABLE 4.3: BFA Results - Dominant Fingerprints

This BFA variation classified 589,402 fragments as text which correspond to the 30.3% of the initial corpus. 490,267 of them are fragments that originate 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 positive rates in identifying document-type fragments as text, with 16.8% false positives. This percentage of true positive rates corresponds to the 65.3% of the total pdf, xls, doc and text files of our corpus.

4.2.4 Variation 4 - Fragments above 75% Plain Text Concentration classified as text

According to the results of Table 4.2 almost all text fragments (99.5%) contain more than 75% of plain text. In the same concentration 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. For all the other file types, in this concentration category belong only a tiny amount of their total fragments. We thought that it would be interesting to apply the BFA of variation 1 only to the fragments which contain less than 75% plain text 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 4.4. 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 pdf, xls, doc and text files of our corpus.

	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
ptc* >75%	23,892	0	188,564	20,691	0	1,308	3,053	1,026	0	0

*plain text concentration

TABLE 4.4: BFA - Fingerprints Trained in 0-75% and tested in 0-75%

4.2.5 Optimal Variation for Text Fragment Classification

In Table 4.5 we present the precision and document-type fragment retrieval percentages of all the 4 BFA variations.

Profoundly enough, the second variation is by far the worst and cannot aid the design process of our classification algorithm. Among the other three variations, variation 4 yields the best results. Both coverage and precision 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% plain text concentration as text is a major weakness.

In a real life scenario, the ratio between the amount of fragments of every file type it's 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 plain text concentration higher than 75% would 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 bigger, renders variation 4 unscalable. We should first conduct similar research for all the existing file types, 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 among the 4 BFA variations that we tested, variation 1 is the optimal regarding text fragment classification .

	precision	document-type fragment retrieval
Variation 1	85	66.7
Variation 2	31.5	11.7
Variation 3	83.2	65.3
Variation 4	86.8	68.3

TABLE 4.5: BFA Variation Comparison

4.2.6 BFA Training - Entire 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 the BFA that uses the entire ASCII byte set is essential, in order to choose which approach is the best for the design of our algorithm.

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
pdf	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

TABLE 4.6: BFA Results - Entire ASCII Byte Set Training

Shahi[19] tested a BFA for fragment classification using the exact same file types as we do. The only exception is that he used the entire 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 of each file type, is approximately the 10% of Shahis training set. In order to have a more objective comparison, we are going to compare the results that Shahi got by using fingerprints that were trained with the 10% of his training set, with our variation 1 BFA. That way, fingerprints from both approaches have almost the same amount of training. The results can be found in Table 4.6.

For broad fragment classification, fingerprints that use the entire byte-set seems to be way more effective than variation 1. Only the accuracies for pdf and ppt fragments are higher in variation 1, simply because Shahis BFA variation achieved 0% of true positive rates for these file types.

Regarding text fragment classification the precision rates are pretty close. We took the accuracy percentages that correspond to text fragment classification from Table 4.6 and calculated the amount of fragments that would have been classified as text using the default BFA. We should mention that since Shahis BFA variation is not limited in only classifying fragments that contain plain text, the amount of fragments that this BFA variation can process is bigger (Table 3.1).

According to this, that BFA would have classified 462,345 fragments as text which correspond 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% of true positive rates in identifying document-type fragments as text, with 11.3% false positives. This percentage of true positive rates 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 precision of Shahis BFA variation is slightly higher (88.7%) from variation 1 (85%), the amount of document-type fragments that are 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 Shahis BFA variation that would have classified 410,173. By using BFA in our algorithm, we aim to retrieve us much pdf, xls, doc and text fragments as possible and minimize false positive rates. In that case, this is a trade-off between precision and the amount of document-type fragment retrieval. Precision levels are pretty close. However, variation 1 classifies significantly more (22%) document-type fragments as text. For that reason, although this estimation is approximate, we chose to use variation 1 over a BFA that uses the complete ASCII byte set for its fingerprints training. Therefore, our final algorithm will make use of BFAs variation 1.

4.3 BFA Text Output Analysis

4.3.1 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 BFAs output and it seems that BFA classified as text fragments with diverse plain text concentration. This analysis can be found in Table 4.7.

plain text concentration	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png
$0 < c \leq 25$	13.9	0.8	0	57.4	29	78.6	36	10.1	1.4	55
$25 < c \leq 50$	34.7	99.1	0.1	18.4	71	19.1	50.3	80.1	98.2	44.9
$50 < c \leq 75$	14	0.1	0.1	1.6	0	1.6	5.1	0.9	0.4	0.1
$75 < c \leq 100$	37.4	0	99.8	22.6	0	0.7	8.6	8.8	0	0

TABLE 4.7: BFA Output Plain Text Concentration Analysis

Although the 85% of BFAs output originates from document-type files, our BFA extension algorithm considers all these fragments to be of document-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 rates 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 BFAs output trying to find patterns that will help us predict the file type of the document-type fragments. We introduce two new metrics,

the Individual Null Byte Frequency (INBF) and the Plain Text Concentration (PTC). The INBF in conjunction with Shannon entropy[20] can be used to effectively distinguish between pdf from xls and doc fragments. Additionally, the PTC metric can be used to eliminate the chances of a fragment, that belongs to a certain plain text concentration category, to be falsely classified.

4.3.2 Plain Text Concentration Categories

As we already mentioned, file fragments of certain file formats are expected to have a characteristic 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, but could also be used with any size multiple of 512. As we already have seen in Table 4.2, 75% or more of text fragments is plain text and the majority of xls fragments (97%) are less than 50% plain text. Moreover, more than 90% of the total mp4, zip, ppt, jpg, png and ogg fragments belong in the 25-50% plain text concentration category. This is quite reasonable taking under account that these fragments originate from compressed file formats of high entropy. 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 reason that this light-weight metric can be combined with current techniques to 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 prediction 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 an mp4, zip or ogg fragment etc.

4.3.3 Shannon Entropy

There is a widespread use of the Shannon entropy[20] metric in file fragment classification techniques. Entropy measures how much information a sequence of symbols contains. 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[4][3]. In figure B.2 we can see the entropy distribution among these file fragments.

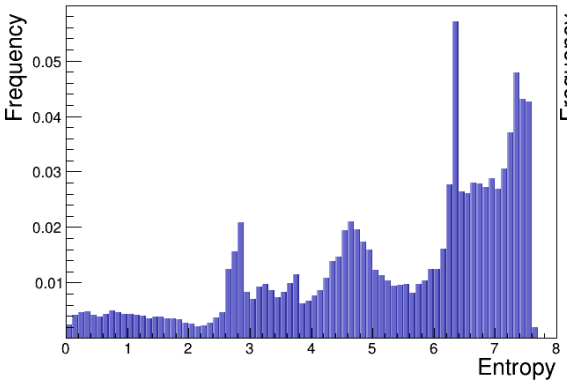

Figure 4.1.A: Pdf distribution

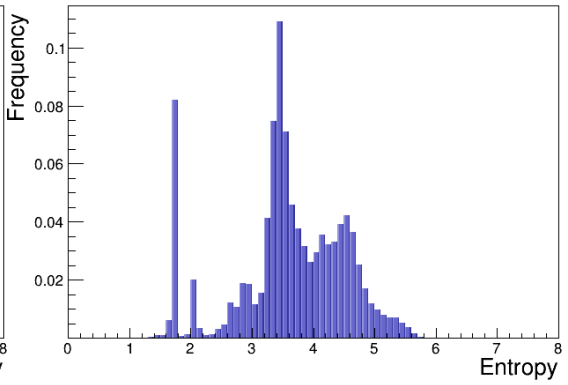

Figure 4.1.B: Xls distribution

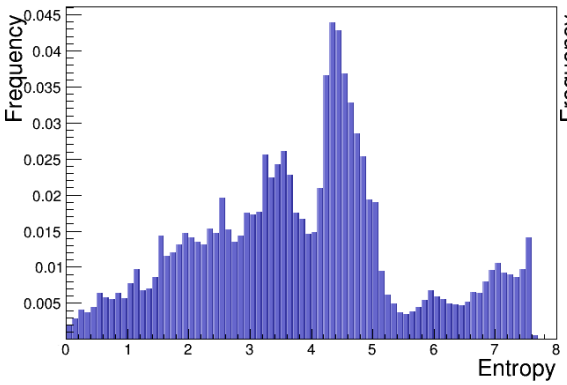

Figure 4.1.C: Doc distribution

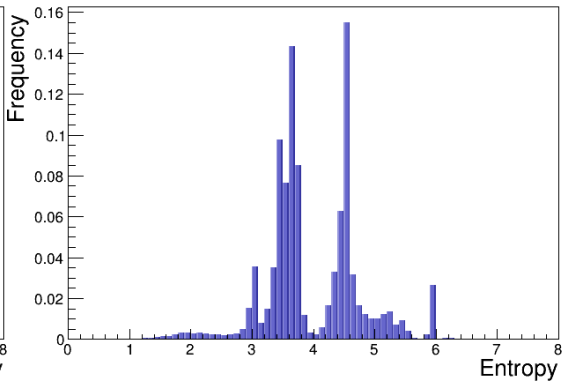

Figure 4.1.D: Text distribution

FIGURE 4.1: Entropy Distribution

As expected, by being a compressed file format pdf have significantly higher entropy values than doc, xls and text fragments. A significant amount of pdf fragments have an entropy value of 6 or more, in contrast with the other file types where the majority of their fragments have an entropy of lesser value. Additionally, only pdf and doc fragments have an entropy value higher than 6.

4.3.4 Individual Null Byte Frequency

We applied several statistical measurements such as median, mean, mode, standard deviation, minimum and maximum frequency byte values in BFAs output fragments. However, we couldn't find strong distinguishable characteristics for these file types that could aid our algorithms design. Thereafter, we manually inspected several fragments of all file types, and noticed that the amount of null bytes in xls fragments was significantly high. However, although slightly less, the frequencies of null bytes were 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 for the xls fragments these sequences were fewer. Additionally, the majority of the total null bytes in xls fragments were individual. Therefore, we analysed the distribution of individual null bytes of all the document-type fragments. As we can see in figure 4.2 the number of individual null bytes in xls fragments is obviously higher compared to 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.

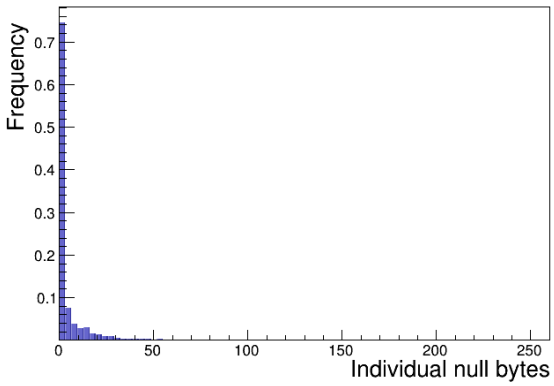

Figure 4.2.A: Pdf distribution

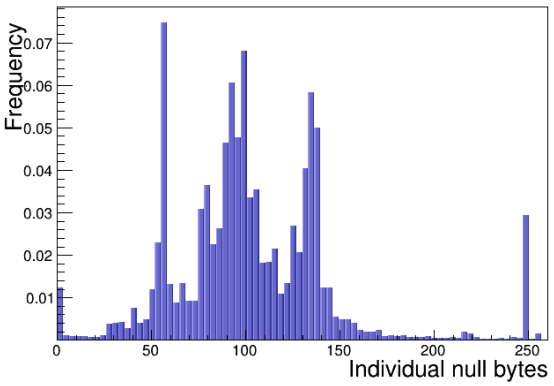

Figure 4.2.B: Xls distribution

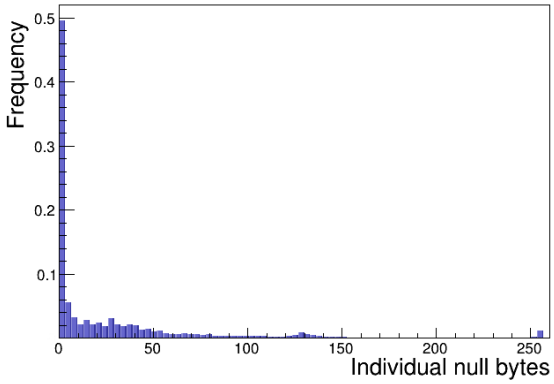

Figure 4.2.C: Doc distribution

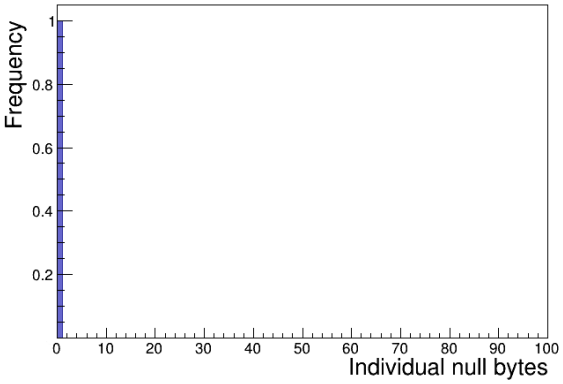

Figure 4.2.D: Text distribution

FIGURE 4.2: Individual Null Byte Distribution

4.4 | 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 [4] 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 for two-group file classification. Even though the dynamic version is faster than the naive approach of the algorithm, with runtime complexity $m \times n$, where m, n the length of the input strings, it still seems like an "expensive" technique to be used for file fragment classification.

What Calhoun did was to extract every longest common subsequence of every file fragment of a certain file format and concatenate them in a big string. This string was 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 doc and 500 fragments of the xls type for our representative string creation. This resulted to $500 \times 500 - 500 = 249,500$ comparisons for each file type in order to extract their longest common subsequences. 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 [19]. However, we also consider our testing set significantly small, but since our goal was to find out what is the correlation between speed and accuracy of that technique, a testing set of that size is sufficient. The results can be found in Table 4.8.

As someone would expect, by using longer strings as file type representatives, the classification precision is enhanced. The precision gradually increases while using longer

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

TABLE 4.8: Longest Common Subsequence comparison - doc VS. xls

strings. Although the precision of this metric proved to be on par with the results Calhoun presented in [4], 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 under-construction algorithms speed, and with rough comparisons, the longest common subsequent technique takes 15% longer 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 tested only 2, it is obvious that the difference in speed is quite significant. Moreover, since the use of PTC and INBF yielded few false positive rates between the xls and doc file-types (Chapter 6, the use of LCS technique could not improve the overall performance of our algorithm. 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.

5 | Algorithm Description

In this chapter we describe the final form of our classification algorithm. For simplicity, we don't describe BFA part since all the information of this algorithm can be found in [14].

We divide our final algorithm in three parts and present it in a pseudocode form. Additionally, we provide a decision tree figure in order to make our algorithm more comprehensible for the reader.

The algorithm that we describe assumes that BFA has already read the byte stream of a fragment, created an accuracy level for each file type and classified it as text.

Algorithm - Part 1 Initial state and value declarations

Require:
float $pdfConValue, docConValue$ ▷ BFAs confidence values
byte[] $byteStream$ ▷ Byte content of fragment

Ensure: $XLS, PDF, DOC, TEXT$ ▷ Classification result

1: **declare integer** $ninb$ ▷ Number of Individual Null Bytes in fragment
2: **declare float** $entropy$ ▷ Entropy of the fragment
3: **declare float** ptc ▷ Plain Text Concentration in fragment

4: **declare const integer** $lowNinb := 9$
5: **declare const integer** $highNinb := 25$
6: **declare const float** $textMaxEntropy := 6$
7: **declare const float** $xlsMaxPtc := 50$
8: **declare const integer** $xlsMinNinb := 50$
9: **declare const float** $medianPdfEntropy := 5.8$
10: **declare const float** $lowEntropy := 3.9$

In Part 1 we can see the initial state of our algorithm and all the constant variable declarations. We reuse only the accuracy levels (confidence values) that BFA produced, which correspond to the doc and pdf file types. We noticed that between pdf and doc fragments, when the pdf accuracy level is higher than doc, most of the times the fragment was of the pdf file format.

In lines 1-3 we declare the variables that hold the values of our classification metrics. These metrics are the Individual Null Byte Frequency (INBF), the Shannon entropy and the Plain Text Concentration (PTC). The only prerequisites to calculate these values is the byte stream of the fragment. Furthermore, the values of the constant variables in lines 4-10 are result of the analysis we conducted in Chapter 4.

The "Ensure" field contains the values that our classification algorithm returns as output. Since our algorithm was intended to be able to classify fragments of the doc, xls, pdf and text file types, the returned values are the names of these file types. So for example if the output is DOC then it means that our classifier classified the input fragment as doc.

Apart from the textMaxEntropy and medianPdfEntropy values, all the other values were initially formulated by the histogram analysis and further calibrated with multiple tests.

Algorithm - Part 2 Auxiliary functions

```
11: function isXls()
12:     return ninb > xlsMinNinb ∧ ptc < xlsMaxPtc
13: end function

14: function isPdf()
15:     return pdfConValue > docConValue ∧ ninb ≤ lowNinb ∨
16:             entropy ≥ medianPdfEntropy
17: end function

18: function isPlainText()
19:     return ptc == 100
20: end function

21: function isProbablyNotPdf()
22:     return entropy ≤ lowEntropy ∧ ninb ≥ highNinb
23: end function
```

In Part 2 we provide all the auxiliary functions that we use in our classifier. All of them evaluate a boolean expression and return a boolean value.

The "isPlainText" function checks if the fragment is fully comprised of byte values that correspond to plain text characters. Moreover, the "isXls" function checks the amount of individual null bytes in the fragment, in conjunction with its plain text concentration. This is due to the fact that fragments of the xls type contain a high number of individual null bytes in contrast with the other 3 file types. Additionally, the majority of xls fragments have less than 50% plain text concentration.

Similarly, the "isPdf" function returns true, either if the entropy of the fragment is higher than the pdf median entropy value, either if the accuracy level that BFA gave is higher than docs in conjuction with low number of individual null bytes. We chose to use the pdf median entropy value instead of the mean, because the histogram analysis revealed a skewed entropy distribution. Median is preferred from mean, as the best measure of central tendency in non-normal distributions.

Finally, the "isProbablyNotPdf" function classifies a fragment as possibly not being of pdf format when the entropy of the fragment is pretty low and the number of individual null bytes relatively high.

Algorithm - Part 3 Classifier

```
24: if isPlainText( ) then
25:     if entropy < textMaxEntropy then
26:         return TEXT
27:     else if isPdf( ) then
28:         return PDF
29:     else
30:         return DOC
31:     end if
32: else
33:     if isXls( ) then
34:         return XLS
35:     else if isProbablyNotPdf( ) then
36:         return DOC
37:     else if isPdf( ) then
38:         return PDF
39:     else
40:         return DOC
41:     end if
42: end if
```

In Part 3 of our classification algorithm we make use of an if-statement decision tree, combined with the aforementioned functions plus some additional checks.

In line 24 we check if the fragment is fully comprised of plain text characters. If it does, then we eliminate the chance of being of the xls file format. In line 25, we check the entropy value of the plain text fragment. If that value is below the maximum entropy value we found for text fragments in BFAs output, we classify that fragment as text. Otherwise, if it has higher entropy then its either pdf or doc. The remaining part of the pseudocode is pretty simple so we won't elaborate further.

Since the multiple nested if-statements hinder readability, we additionally provide figure 5.1 that presents our classification algorithm as a decision tree .

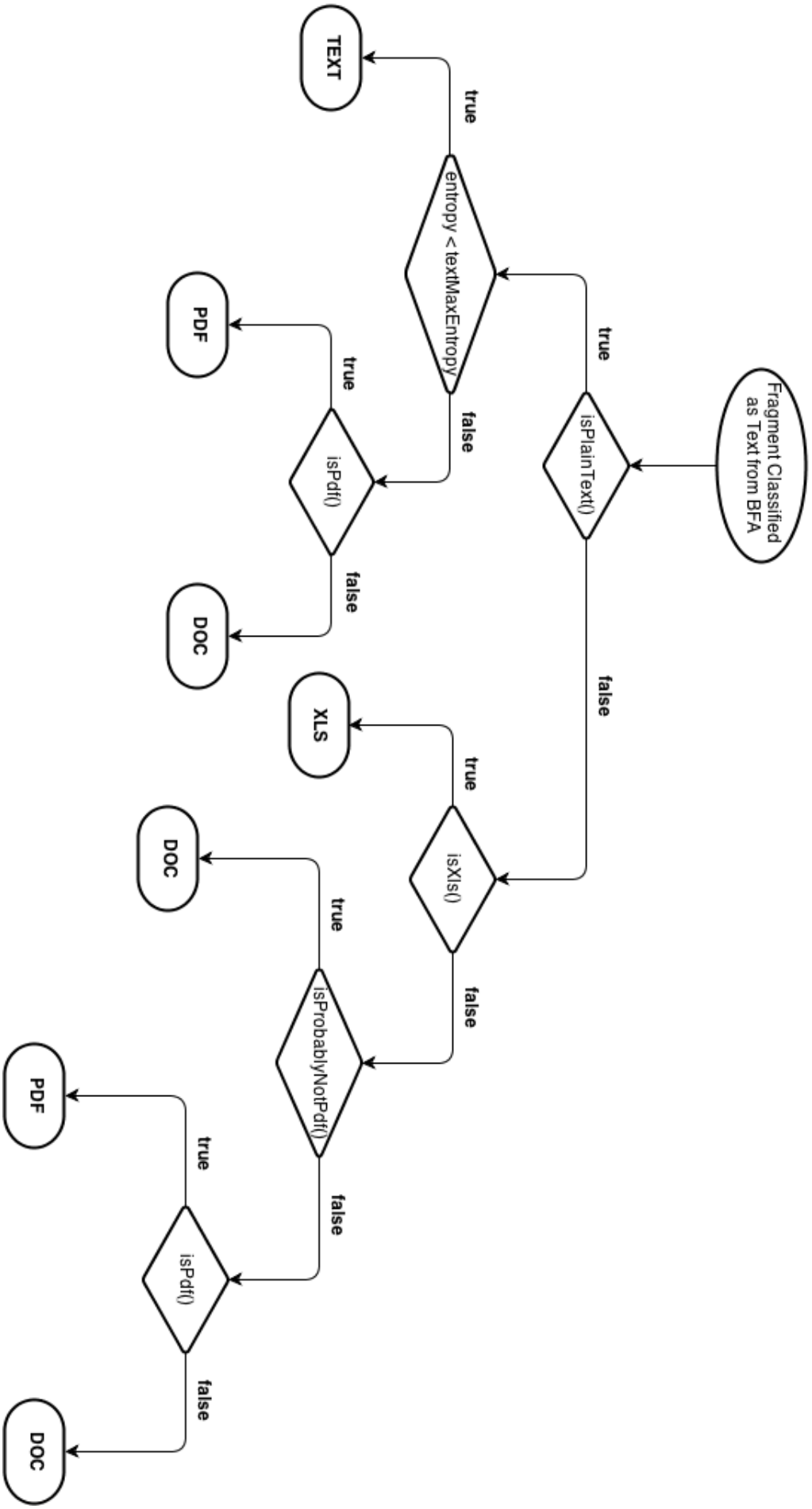

FIGURE 5.1: Algorithm as Decision Tree

6 Results

In this chapter we present the accuracy results of our classification algorithm. Analysis of the results will be presented in the next chapter. Since our algorithms design is based on the analysis we conducted on the experimental data set (Table 3.1), the algorithm is biased towards this set. For that reason we used the final testing data set (Table 3.2) to test our algorithms performance.

Although our final algorithm was implemented in one piece, we divide the results in three parts. In section 6.1 we provide the performance of the BFA part of our algorithm regarding document-type fragment classification. In section 6.2 we provide the classification results of the algorithm we described in Chapter 5. We should note that the classification results of that part correspond to the data set of fragments that were initially classified as text from BFA. Finally, in section 6.3 we provide the overall classification results of our classifier. We also provide an overview of the complete algorithm in Figure 6.1. The numbers on the figure correspond to the respective sections of this chapter.

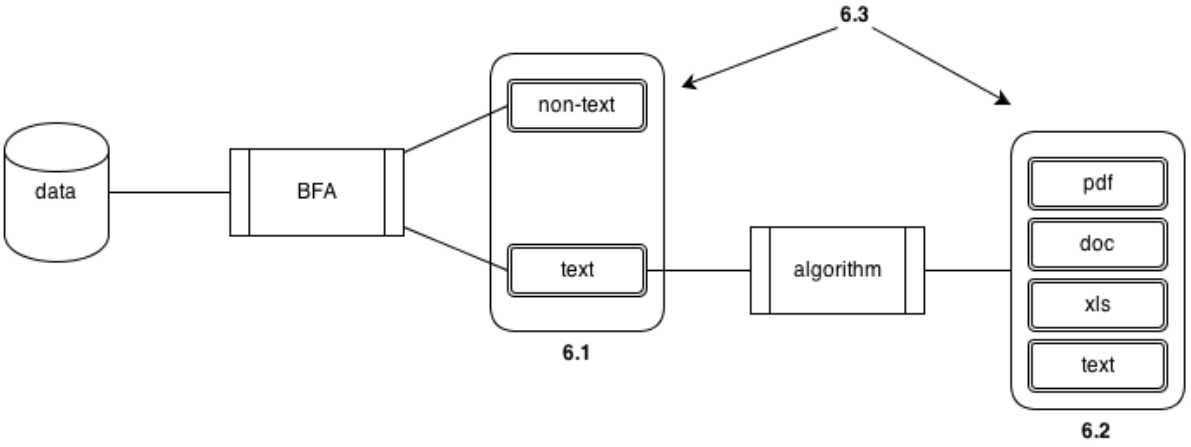

FIGURE 6.1: Algorithm Overview

6.1 BFA Scan - Text Fragment Classification

In Table 6.1 we present the performance of BFA part of our algorithm regarding text fragment classification. The first row corresponds to the initial number of fragments our algorithm processed for each file type. The other rows provide information regarding BFAs document-type fragment classification rates.

	pdf	text	doc	xls	ppt	mp4	ogg	zip	png	jpg
num. of fragments	1,874,910	1,891,472	1,775,747	1,870,376	1,864,145	1,888,605	1,891,754	1,889,477	1,880,742	1,886,853
fragments classified as text	385,616	1,889,662	950,308	1,756,200	404,198	67,107	111,738	94,815	86,017	98,960
%	20.6	99.9	53.5	93.9	21.7	3.6	5.9	5	4.6	5.2
fragments classified as other	1,489,294	1,810	825,439	114,176	1,459,947	1,821,498	1,780,016	1,794,662	1,794,725	1,787,893
%	79.4	0.1	46.5	6.1	78.3	96.4	94.1	95	95.4	94.8

TABLE 6.1: BFA Text Fragments Classification

6.2 BFA Extension Algorithm Accuracy

In Table 6.2 we provide the accuracy results of our custom algorithm as a percentage confusion matrix. The columns represent the actual type of the fragments while the rows represent the file type that the fragments was classified as. Since our algorithm was designed to classify fragments of the doc, pdf, xls and text file formats, we only have true positive rates for these 4 file types. We present the true positive rates as shaded cells.

	pdf	text	doc	xls	ppt	mp4	ogg	zip	png	jpg
fragments classified as text from BFA	385,616	1,889,662	950,308	1,756,200	404,198	67,107	111,738	94,815	86,017	98,960
pdf	46.3	0.5	16.2	2.3	31.2	87.6	95.5	90.6	84.9	83.0
text	38.9	98.8	11.7	3.9	1.2	0	0.1	0	2.2	6.4
doc	13.6	0.7	60.7	17.9	43.4	12.4	4.4	9.3	8.8	8.7
xls	1.3	0	11.4	75.9	24.2	0.1	0	0.2	4.1	2

TABLE 6.2: BFA Extension Algorithm Accuracy

6.3 | Complete Algorithm Accuracy

In Table 6.2 we provide the accuracy results of our complete algorithm as a percentage confusion matrix. The columns represent the actual type of the fragments while the rows represent the file type that the fragments was classified as. Our algorithms can make 5 classification decisions. These decisions can be doc, pdf, xls, text and other. In the "other" broad classification category belong every fragment that was classified was as ppt, ogg, mp4, zip, png or jpg. We present the true positive rates as shaded cells.

	pdf	text	doc	xls	ppt	mp4	ogg	zip	png	jpg
num. of fragments	1,874,910	1,891,472	1,775,747	1,870,376	1,864,145	1,888,605	1,891,754	1,889,477	1,880,742	1,886,853
pdf	9.5	0.5	8.7	2.2	6.8	3.1	5.6	4.5	3.9	4.4
text	8.0	98.8	6.3	3.7	0.2	0	0	0	0.1	0.3
doc	2.8	0.7	32.5	16.8	9.4	0.4	0.3	0.5	0.4	0.5
xls	0.3	0	6.1	71.3	5.3	0	0	0	0.2	0.1
other	79.4	0.1	46.5	6.1	78.3	96.4	94.1	95.0	95.4	94.8

TABLE 6.3: Algorithm Accuracy Results

7 Analysis

In this chapter we analyse the results of Chapter 6. We divide this chapter in three sections. In section 7.1 we analyse the performance of BFA and the amount of fragments that were classified as text. In section 7.2 we analyse the performance of our complete algorithm. Finally in section 7.3 we compare our algorithm performance with the classification algorithms that Shahi tested in [19].

7.1 BFA performance

Judging from the results of Table 6.1, the BFA part of our algorithm performed as expected. It yielded almost identical results with the results that we got during the algorithms development procedure (Chapter 4).

In a corpus of 18,714,081 fragments BFA classified 5,844,621 fragments as text. From that amount, 4,981,786 fragments do originate from document-type file formats and 862,835 from the other 6 file types. This means that we have a percentage of 85.2% true positive rates and a 14.8% of false positives, regarding document-type fragment classification. These results are on par with the ones we got when we used this variation of BFA in a ten-times smaller corpus (Table 4.1).

Moreover, this 85.2% of true positive rates correspond to the 67.2% of the total doc, pdf, xls and text fragments of our initial corpus. However, this percentage is not representative for a real life case, since our final testing set was comprised only of fragments that had at least one plain text character. Considering that we know the percentage of fragments with no plain text (Tables 3.1, 3.2) for each of the 10 file types, we can calculate the approximate overall document-type fragment retrieval in a corpus of 10GB size. Thus, taking under account the amount of fragments that correspond to fragments with 0% plain text concentration, BFA would have approximately retrieved the 63% of the total doc, pdf, xls and text fragments from a 10GB corpus.

7.2 | Overall Algorithm Performance

The accuracies of the 4 document-type file formats of our interest are quite encouraging. We got 9.5% for pdf, 98.8% for text, 32.5% for doc, 71.3% for xls fragments and 92.4% for classifying a fragments as of another file type. It seems that due to the high entropy of the non-document type fragments, most of the fragments that were initially falsely classified as text from BFA, thereafter were falsely classified as pdf. This is not necessarily a bad thing, since it kept false positives rates for the doc, xls and text file types at low levels. Especially for the text and xls file types the false positive rates are extremely small.

However, the information that we can get from this results are insufficient to evaluate our algorithms performance. In addition, this is the main weakness of the scientific publications in the file type validation algorithm field. We observed that most of the papers upon file type validation techniques provide only the true positive rates and do not give additional information regarding the classification capacity of their algorithms. Surprisingly enough, this is in general a common phenomenon in algorithm comparison studies [6][21].

We consider that by calculating the $F-score$ along with the overall accuracy of our algorithm we can acquire better insight for our algorithms performance. The $F-score$ is a statistical measure that tests accuracy and it considers both the precision and recall measures of the test to compute the score. A higher $F-score$ implies higher accuracy. The $F-score$ measurement is described in formula 7.3 . Note that tp stands for "true positive", fp for "false positive", tn for "true negative" and fn for "false negative".

$$precision = \frac{tp}{tp+fp} \quad (7.1) \qquad recall = \frac{tp}{tp+fn} \quad (7.2)$$

$$F-score = 2 * \frac{precision * recall}{precision + recall} \quad (7.3)$$

Additionally, the equation that we use to calculate the overall accuracy of our classifier can be found in formula 7.4.

$$overall\ accuracy = \frac{correct\ predictions}{total\ predictions} \quad (7.4)$$

The results of the aforementioned measurements can be found in Table 7.1. Our classifier performed extremely well for the text and xls file types as well as for the broad non-document-type classification class with an $F - score$ of 0.91, 0.78 and 0.86 respectively. The prediction rates for the doc file type are quite satisfying with an $F_1 - score$ of 0.39. We say quite satisfying considering that the doc file type was the most "tricky" among the document-file types we analysed, due to the fact that we couldn't find strong distinctive characteristics. Furthermore, the worst rates are for the pdf file type, mainly because BFA classified as text only the 20% of the initial pdf fragments. Additionally, during the second phase of our classification algorithm most of non-document type fragments were falsely classified as pdf due to their high entropy. Finally, the overall accuracy of our classifier is 0.77, a very high number for a file carving technique. This means that in 100 predictions, 77 of them will be correct. Even if we exclude the classification rates of BFA that classifies fragments in a broad class of non-document fragments, the overall accuracy of our classifier for the pdf, doc, xls and text file types remains high with an accuracy of 67%.

	pdf	text	doc	xls	other
Precision	0.1	0.85	0.49	0.86	0.81
$F - score$	0.1	0.91	0.39	0.78	0.86
Overall accuracy:	0.77				

TABLE 7.1: Algorithm Prediction Rates

7.3 | Algorithm Comparison

In this section we try to compare the performance of our classifier with similar classification techniques. In subsection 7.3.1 we compare the accuracies of the algorithms and in subsection 7.3.2 we compare their performance in terms of execution time.

Among the algorithms we compare are the Byte Frequency Analysis algorithm [14], the n-Gram Analysis [12], the Rate of Change [11] and the algorithm of Conti et al. [5]. We use the results that Shahi acquired from his experiment [19]. Moreover, we chose to compare our algorithms performance with Shahis results, because he used the same 10 file types for his experiment. Additionally, his testing set was of the exact same size as the one we used for our final testing set. One of the major problems in the file carving field is that there is no proper comparison between classification algorithms, since every

technique was tested in different data sets that were comprised of different file types. Furthermore, at this point we should note that for the comparison of the algorithms, we use the results that Shahi got by using fingerprints that were trained with the 10% of his total training set. This 10% corresponds to the size of the training set that we used to train our fingerprints.

7.3.1 Accuracy Comparison

Although the testing sets that were used for the experiments are similar, it is still quite difficult to compare these algorithms. Our algorithm was specifically designed to classify only fragments of 4 file types or to classify as "other" everything that was classified as non-text from its BFA part. Moreover, all algorithms do not give an $F-score$ for every file type. The results can be found in Table 7.2.

	pdf	zip	text	doc	mp4	xls	ppt	jpg	ogg	png	other	overall accuracy
Our algorithm	0.1	-	0.91	0.39	-	0.78	-	-	-	-	0.86	0.77
Byte Frequency Analysis	-	0.42	0.59	0.01	0.25	0.54	-	0.16	0.17	0.17	-	0.33
Rate of Change	0.37	-	0.73	0.5	-	0.8	0.22	-	-	-	-	0.32
N-Gram Analysis	0.17	-	0.89	0.12	-	0.74	0.22	-	-	-	-	0.30
Algorithm of Conti et al.	0.10	0.46	0.44	0.16	0.37	0.38	0.06	0.23	0.16	0.08	-	0.30

TABLE 7.2: Classification Algorithm Accuracy Comparison

Our algorithm outperforms the other 4 in classifying fragments of the text file format with an $F-score$ of 0.91. Furthermore, it comes second only after the Rate of Change algorithm regarding xls and doc fragment classification. Lastly, the $F-score$ for the pdf file type is the worst along with the algorithm of Conti et al. Moreover, our algorithms overall accuracy is approximately 2.5 times higher than the overall accuracies of the other algorithms. However, we should not forget that our algorithm is limited in classifying only fragments that contain at least one plain text character. In general, although our algorithm takes less decisions and classifies non-document fragments in a broad class, its decisions are significantly more reliable than the decisions of its competitors.

7.3.2 Execution Time Comparison

In this subsection we provide the average execution time of our algorithm and we compare it with the results that Shahi provides in [19]. We define the average execution time as the amount of time that it takes to classify a set of fragments divided by the

total amount of fragments. Although we used the Java programming language for all our implementations while Shahi used C++, we could make an approximate comparison of their performance. Additionally, the machine in which we benchmarked our algorithm has similar specification with the one that Shahi used but its not the same. We benchmarked our algorithm in a 64-bit OS with an Intel Core i7-2600 @ 3.40 GHz×8 processor, 15.6 GB of RAM and a 1.1TB size hard drive (7200 RPM 32MB Cache SATA 3.0Gb/s).

algorithm	avg. execution time(in seconds)
N-Gram Analysis	0.00041
Our algorithm	0.001651
Byte Frequency Analysis	0.004344
Rate of Change	0.049941
Algorithm of Conti et al.	0.085005

TABLE 7.3: Classification Algorithm Average Execution Time

As we can see in Table 7.3, our algorithm is ranked second regarding runtime performance. N-Gram analysis is the fastest technique among the 5 algorithms, outperforming our algorithm by a magnitude of 4. Although our algorithm makes use of a BFA variation in combination with some additional metrics, it is approximately 3-times faster than the BFA variation that Shahi used for fragment classification. This is probably due to the fact that our algorithm uses only 98 byte values that correspond to plain text characters. This means that when our algorithm calculates an accuracy level of a file type using the respective fingerprint, it only has to calculate 98 values. A BFA that analyses the entire ASCII byte set requires 256 calculation in order to produce an accuracy level. Moreover, our algorithm outperforms Rate of Change and Conti et al. algorithms by a magnitude of 30 and 50 respectively.

As we already mentioned, these comparisons are approximate. We can only say with certainty that our algorithms runtime performance is slower than n-Gram analysis and faster than the other three algorithms. Furthermore, with rough estimations, it will take around 1 hour for our algorithm to analyse 1GB of fragments.

8 Discussion

In our experiment, during the algorithm development procedure our analysis yielded two new classification metrics. The Individual Null Byte Frequency (INBF) and the Plain Text Concentration (PTC). The representative INBF values that we used to classify mainly document-type fragments were formed from BFAs output. The output was comprised only of fragments that were classified as text. This resulted in a disproportionate set of fragments for every file type. For instance, the amount of pdf and doc fragments we analysed were significantly less than the amount of text and xls. The same holds also for the non-document fragments. Although INBF seems to be quite effective as a part of our classification algorithm, we believe that a more extensive analysis must be made in a bigger data set, in order to be able to say if this metric could be used in other broad fragment classification techniques. On the other hand, PTC analysis was made in a corpus of 10GB in total and we believe that our analysis is consistent and can be a great asset in file carving techniques.

Although we did our best to eliminate possible biases in our experimental setup, we can not guarantee the integrity of our corpus. Considering that our corpus was comprised of tens of thousands of files it wasn't feasible to manually check every file. We can't say for sure if the suffixes of every type correspond to the actual file format. We did some manual inspections in the experimental data set and we found and removed about 37MB of files that had a *.txt* suffix but weren't text files. Additionally, we don't know if our corpus was comprised of a single or of various file format versions. For example, an Adobe PDF 1.7 document might be slightly different from an Adobe PDF 1.3 document.

Moreover, we are aware that our results correspond only to our controlled corpus. Considering the big number of file format that is available, there is a possibility that files of different formats might have similar characteristics with the document-type fragments we analysed. In addition, taking under account a realistic forensic scenario, where the number of different file formats is most likely to be higher (e.g 100 file types), the outcome may significantly vary.

Furthermore, in a corpus where the amount of fragments of different file types is not of the same ratio, the performance of our algorithm is expected be different. We strongly believe that this would be the case especially for the predictions of pdf and doc file fragments, since we couldn't find very strong distinguishable characteristics. Conversely, the prediction capabilities of our classifier regarding text and xls file fragments won't vary too much even in a non 1:1 corpus. We found that fragments of the xls file type contain a high number of individual null bytes and their plain text concentration is below 50%. A possible explanation for this behaviour is that the high number of null bytes is due to the cell structure of xls sheets. In addition, since this number is pretty high and in conjunction with the fact that xls sheets are being used mainly as a calculation tool than writing voluminous texts, the concentration of plain text remains at a low level.

Lastly we expected that text fragments would be fully comprised of plain text. This was verified from the analysis we did during the algorithm development procedure, but there were a tiny percentage (2%) of fragments that contained less plain text concentration. This confirms our concerns regarding our corpus integrity and although its a negligible percentage of the total text fragments, it is present, making us slightly sceptical towards the other files types of our corpus.

9 | Conclusions and Future Work

In this chapter we will summarize the conclusions we reached and give directions for future work.

9.1 | Conclusions

In this project we created a file fragment classifier for document-type fragments. We used a large data set of about 20GB size which contained files from 10 different file formats. We made use of the a variation of Byte Frequency Analysis algorithm to classify document-type fragments in a broader class. After this point we created a custom algorithm in order to be able to distinguish between the files that were classified initially as text from the BFA. Our results show that the BFA variation that we used is quite effective in distinguishing between document-type fragments from other file formats.

Additionally, we introduced two new classification metrics. The Individual Null Byte Frequency (INBF) and the Plain Text Concentration (PTC). The INBF metric can be used to enhance xls fragments classification due to the fact that fragments of that file type have a significantly higher amount of null bytes compare to the fragments of the other 9 file types that we used. Moreover, the PTC metric is a very interesting finding. Although out of hindsight it looks obvious that files from specific file types will have characteristic plain text concentration, we couldn't find any reference in the existing bibliography. Only Roussev and Garfinkel [18] mention that text files are expected to contain high amounts of printable ASCII characters. We are confident that this extremely light-weight metric can be combined with current techniques and improve their classification accuracy.

In order to evaluate the performance of our classifiers we compared it with 4 different classification algorithms. As it seems our classifier is significantly more accurate with the decisions that it takes achieving an overall accuracy of 77%. It did extremely well in classifying fragments of the text and xls file formats along with fragments of the non-document types, achieving an $F - score$ of 91%, 75% and 86% respectively.

Furthermore, we observed that most of the studies upon the file type validation field do research on techniques that are quite expensive. In addition, we experimented with the Longest Common Subsequence technique which was firstly introduced to the field by Calhoun [4]. We concluded that although this technique is extremely accurate for two-group classification, it cannot be used for broad fragment classification as Calhoun suggests, due to its high runtime complexity.

Lastly, the availability of a multitude of different file types in combination with the newly introduced file formats along with the accuracy and speed requirements of the forensic cases, renders file fragment classification a wicked problem [17]. We achieved quite good performance, in both accuracy and runtime performance, compared to already existing techniques. Our current algorithm can approximately classify 25GB of data per day. However, our current implementation only uses 10 fingerprints, one for each file type. This means that by adding more fingerprints of different file formats the runtime complexity of our algorithm will proportionally rise. Nevertheless, fragment classification has received little attention the past couple of years and we are confident that there is plenty of room for improvements.

9.2 | Future Work

Although the results we got are promising, we consider the classification algorithm we developed a proof-of-concept rather than a practical technique. First of all, we want to use more file types in future experiments and train our fingerprints with more data. More specifically, we would like to analyse more document-type file and analyse their Individual Null Byte and the Plain Text Concentration values. Additionally, if our variation of BFA performs the same regarding document-type classification in a corpus of more file types, we can extend our classifier to further classify the non-document type fragments, which currently are being classified in a broad file class.

This page intentionally left blank.

A | BFA Variation 2 - Results

	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 A.1: BFA - Fingerprints Trained in 0-25% and tested in 0-25%

	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

TABLE A.2: BFA - Fingerprints Trained in 25-50% and tested in 25-50%

	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 A.3: BFA - Fingerprints Trained in 50-75% and tested in 50-75%

	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

TABLE A.4: BFA - Fingerprints Trained in 75-100% and tested in 75-100%

B BFA Output - Histogram Analysis

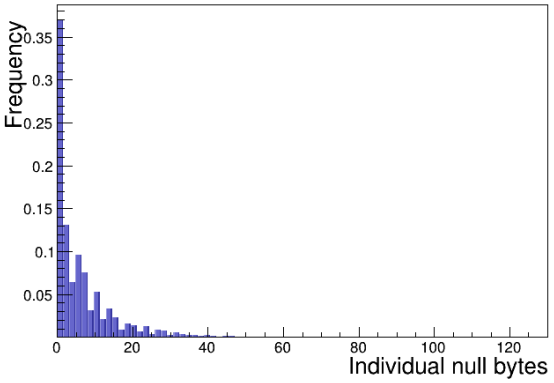

Figure B.1.A: Mp4 distribution

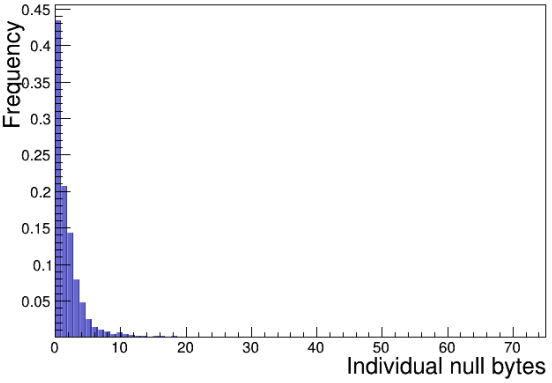

Figure B.1.B: Zip distribution

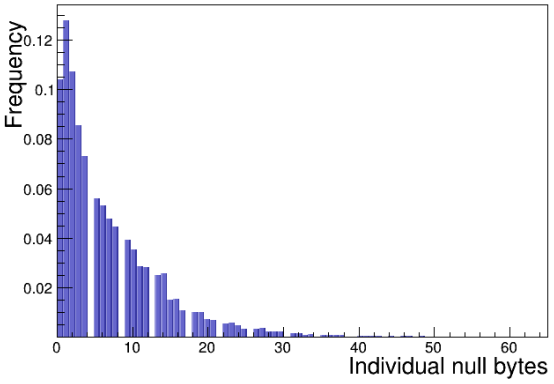

Figure B.1.C: Ogg distribution

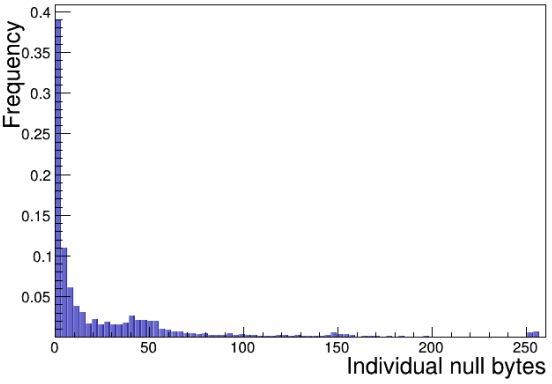

Figure B.1.D: Ppt distribution

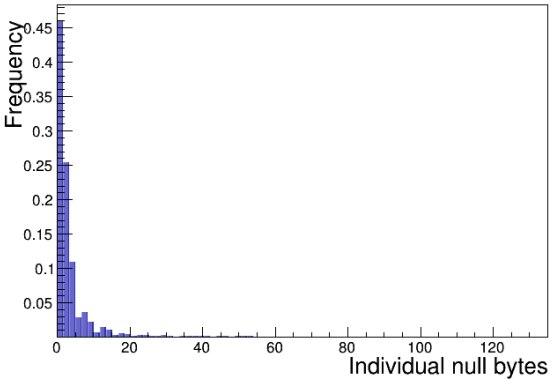

Figure B.1.E: Png distribution

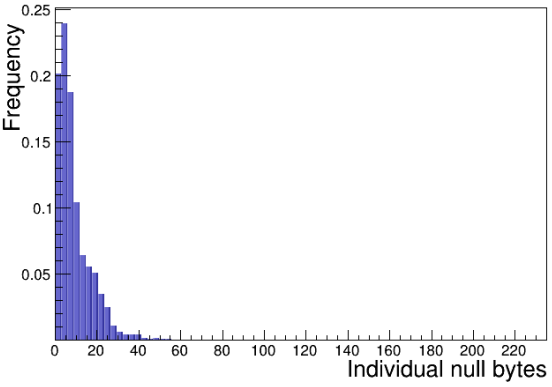

Figure B.1.F: Jpg distribution

FIGURE B.1: Individual Null Byte Distribution

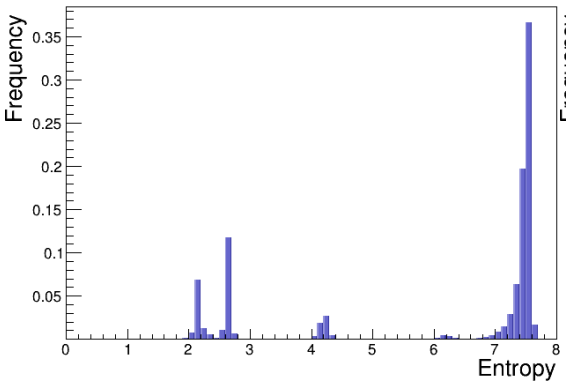

Figure B.2.A: Mp4 distribution

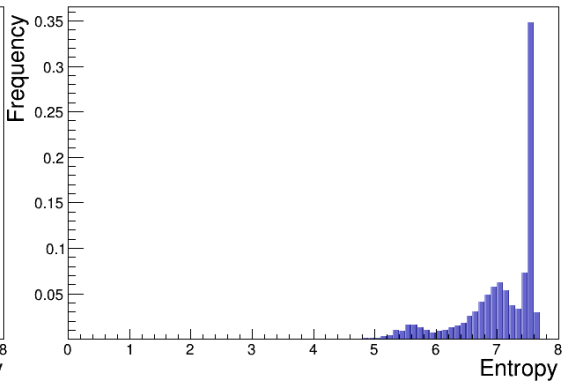

Figure B.2.B: Zip distribution

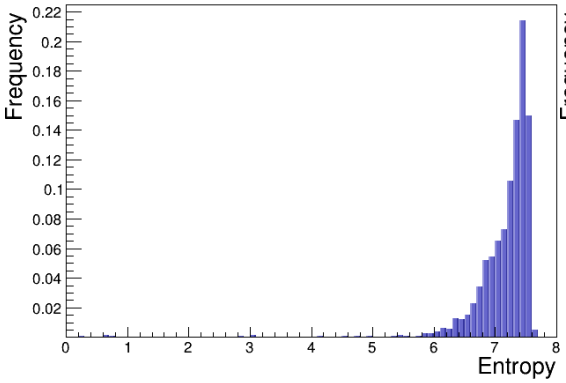

Figure B.2.C: Ogg distribution

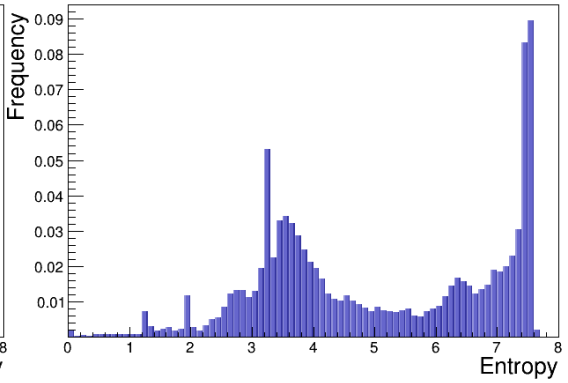

Figure B.2.D: Ptt distribution

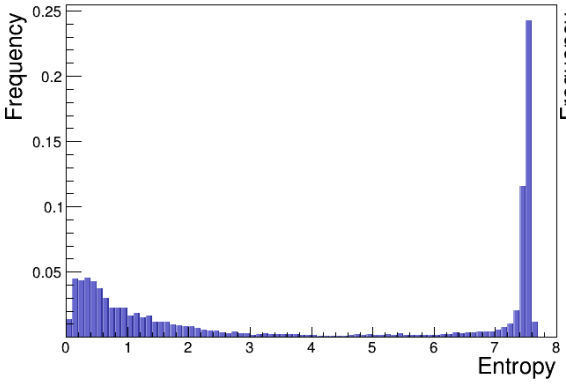

Figure B.2.E: Png distribution

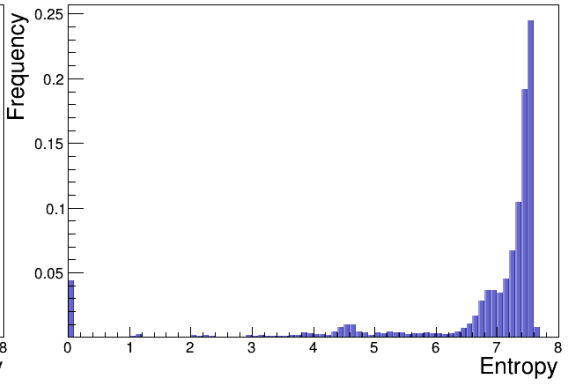

Figure B.2.F: Jpg distribution

FIGURE B.2: Entropy Distribution

Bibliography

[1] Aronson, L., & Bos, J. Van Den. (2011). Towards an Engineering Approach to File Carver Construction. 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, p. 368-373.

[2] Axelsson, S. (2010). The Normalised Compression Distance as a file fragment classifier. Digital Investigation, 7, S24-S31. doi:10.1016/j.diin.2010.05.004

[3] J. van den Bos and T. van der Storm, "Bringing Domain- Specific Languages to Digital Forensics", in Proceedings of the 33rd ACM/IEEE International Conference on Software Engineering (ICSE'11), vol. 2. ACM, 2011.

[4] Calhoun, W. C., & Coles, D. (2008). Predicting the types of file fragments. Digital Investigation, 5, S14-S20.

[5] Conti G, Bratus S, Sangster B, Ragsdale R, Supan M, Lichtenberg A, et al. Automated mapping of large binary objects using primitive fragment type classification. In: Proceedings of the 2010 Digital Forensics Research Conference (DFRWS); 2010.

[6] Demsar, J. 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7:1-30.

[7] Fitzgerald, S., Mathews, G., Morris, C., & Zhulyn, O. (2012). Using NLP techniques for file fragment classification. Digital Investigation, 9, S44-S49.

[8] Garfinkel, S. L. (2007). Carving contiguous and fragmented files with fast object validation. Digital Investigation, 4, pp. 2-12.

[9] S. Garfinkel, P. Farrell, V. Roussev and G. Dinolt, "Bringing science to digital forensic with standardized forensic corpora," in DFRWS, 2009.

[10] Gopal S, Yang Y, Salomatin K, Carbonell J. Statistical learning for file-type identification. In: 2011 10th International Conference on Machine Learning and Applications and Workshops (ICMLA), Vol. 1; 2011. p. 68-73.

[11] M. Karresand and N. Shahmehri, "File Type Identification of Data Fragments by their Binary Structure," in Proceedings of the 7th Annual IEEE Information Assurance Workshop, 2006.

[12] Li W, Wang K, Stolfo S, Herzog B. Fileprints: identifying file types by n-gram analysis. In: IEEE information assurance workshop, 2005.

[13] Maddox, L., & Beebe, N. (2012). Systematic Classification Engine and Data Analysis Overview by Dr . Nicole Beebe.

[14] M. McDaniel and M. Heydari , "Content based file type detection algorithms" , in Proc. 36th Annu. Hawaii Int. Conf. System Sciences (HICSS'3)-Track 9 , IEEE Computer Society , Washington , D.C. , 2003 , p. 332.1

[15] Pal, A., & Memon, N. (2009). The evolution of file carving. Signal Processing Magazine, IEEE, (March), p. 59-71.

[16] G. G. Richard , III and V. Roussev , "Scalpel: A frugal, high performance file carver" ,in Proc. 2005 Digital Forensics Research Workshop (DFRWS) , New Orleans , LA , Aug. 2005.

[17] Rittel, Horst W. J.; Melvin M. Webber (1973). "Dilemmas in a General Theory of Planning". Policy Sciences 4: 155-169. Retrieved 25 April 2013.

[18] V. Roussev and S. L. Garfinkel, "File Fragment Classification - The Case for Specialized Approaches. In Proceedings of SADFE, p. 3-14, 2009.

[19] Shahi, A. Classifying the classifiers for file fragment classification, August 2012.

[20] Shannon CE. The mathematical theory of communication. Bell System Tech J 1948;27:379-423, 623-56.

[21] Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. AI 2006: Advances in Artificial Intelligence Lecture Notes in Computer Science Volume 4304, 2006, pp 1015-1021.

[22] C. Veenman, "Statistical Disk Cluster Classification for File Carving," in Proceedings of the First Internation Workshop on Computational Forensics, 2007.

[23] "Academic Earth," [Online]. Available: http://www.academicearth.org/.

[24] "Digital Corpora," [Online]. Available: http://digitalcorpora.org/.