UNIVERSITY OF AGDER

# Clustered File Type Identification

**John Daniel Evensen**

**Supervisor**
Morten Goodwin

*This master's thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.*

University of Agder, 2015
Faculty of Engineering and Science
Department of Information Systems

**Abstract**

This thesis examines the possibility of expanding the current field of research for file type identification in Digital Forensics. A proposed solution is presented where unsupervised clustering and supervised classification are combined. The experimentation of the proposed solution increases the speed of file type identification, however with a decrease of total identification accuracy. A technique of unsupervised continuous learning is also presented, effectively making the proposed solution capable of adapting to the environment by learning from the test data while performing file type identification. In the best case scenario, identification accuracy increases from 85.8% to 90.4% when using the unsupervised continuous learning technique.

# Preface

The work in this thesis is performed as part of the Master's Programme in Information and Communication Technology for the University of Agder, Grimstad. This thesis is performed under the supervision of professor Morten Goodwin, with some feedback from professor Ole-Christoffer Granmo.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Digital forensics is a field of great importance for the opposition of computer crime. It is often the case that digital forensics teams retrieve data from criminal investigations. This can be in the form of hard disk drives or other storage mechanisms potentially storing sensitive data which can be used to assess the justification of crime. These storage devices may contain large amounts of data, and sometimes even corrupted or deleted data. The digital forensics teams are then tasked with the retrieval of any and all data from these devices, which may prove a significant task if the data is not normally readable.

In digital forensics there exists various techniques of file type identification used for reconstruction of lost data. These are mainly the *extension based*, *signature based* and *content based* techniques. The content based method refers to the identification of file types only by looking at the binary contents of a file, without type specific information such as file headers. This technique has been widely applied and researched in combination with traditional machine learning approaches using supervised learning. This approach has reported great results in terms of identification accuracy.

One of the major difficulties with this approach is that classification algorithms

mostly operate slowly, i.e it can take a long time for classification to complete depending both on the quantity of data and classification algorithm. This can be a problem where classification of data is time sensitive. This is often the case in digital forensics, as the retrieved data can consist of multiple hard disk drives with multiple terabytes each.

This thesis aims to counteract this problem by applying clustering as a technique to classify data at a faster speed. This technique utilized the fact that clustering of data is less costly than that of a full-fledged classifier. This approach also adds a method of unsupervised continuous learning for classification. Instead of building a predictive model based on new data, clusters can simply be extended using this data, presumably resulting in an increase of positive predictive results.

By applying the field of Clustering to file type identification, this thesis aims to investigate the differences in terms of speed, accuracy, and scalability compared to that of traditional approaches. The possibility of unsupervised continuous learning is also investigated.

This chapter introduces the problem of classification using clustering with application to file type identification.

## 1.1 Research Problem

The goal of this thesis is to investigate the possibility of using clustering for file type identification in digital forensics. The main reason for this is to increase the speed of classification in the field of digital forensics where traditional classification techniques are not fast enough to be practically usable.

The outline of this approach is as following. Clustering is performed on the data to be classified, assigning it to clusters which have been classified using the training data. Then, the performance of the implementation is compared to the performance of traditional classification approaches applied within the same field.

This is extended to the investigation of unsupervised continuous learning. Continuous learning refers to the possibility of incremental learning of the test data. This technique is then taken further to establish whether it can be used to identify new file types as they are introduced.

This research problem yields three different research questions to be explored for the proposed solution in this thesis:

- How is the measurements of accuracy, speed and scalability compared to traditional approaches?
- How well does the proposed solution perform by continuously learning it with test data?
- How does it adapt to the introduction of new file types as test data?

### 1.1.1 Measurements

The performance measurements in this thesis are measured in terms of speed, accuracy and scalability. One important thing to note is that the measurement of accuracy within digital forensics is not always the same as accuracy within classification. In digital forensics, accuracy is often measured in terms of correctly identified file types out of all possible files. Therefore, the term accuracy is used herein as the percentage of correctly assigned files. Scalability refers to the performance in terms of speed and accuracy with a varying number of files.

### 1.1.2 Limitations

The proposed solution requires the use of both a classification algorithm and a clustering algorithm. To compare how each combination of all possible algorithms work together for this thesis would be a too significant task. Therefore,

a set of classification algorithms are measured in terms of accuracy, and the best performing algorithm is brought forth and considered as the classifier of choice for the proposed solution. As for the clustering algorithm, only one is considered because of reasons mentioned later in this thesis. This limits the thesis in terms of possible combinations of algorithms, only proving its viability theoretically, not finding the optimal classification or clustering algorithm to possibly increasing the overall results achieved.

## 1.2 Contributions

Extensive research has been done within the file type detection field of digital forensics. The work is performed and published by experimenting with common supervised learning classifiers (See Table 3.1). These classifiers work by first "training" them with a separate learning data set. Using the learned information, these classifiers can predict the type (class) of new data by comparing given features to that of which is learned from training. This approach follows a strict formula; prediction is dependant of training.

It is assumed that the addition of a clustering technique within the field of digital forensics will greatly improve the speed of file type detection as the need to classify each instance of test data is eliminated.

The application of an unsupervised continuously learning technique is also assumed to add to the existing body of research for file type detection. This due to the fact that no research has been found for this field regarding unsupervised learning.

This approach may also generally contribute to pattern detection problems previously only resolved using traditional classification approaches.

## 1.3 Thesis outline

Chapter 2 presents the necessary background of this thesis, giving an introduction to digital forensics, files, and file type identification, as well as the principles of clustering.

Chapter 3 shows previous works and techniques applied to the field of file type identification.

Chapter 4 presents the proposed solution for this thesis.

Finally, Chapter 5 shows the different experiments performed by the proposed solution to establish the validity of the various research questions stated above, while Chapter 6 finally concludes this thesis and its results.

# Chapter 2

# Background

This chapter presents the necessary background for the proposed solution in this thesis. The field of digital forensics is first presented with the intent of motivating why the work in this thesis is important. Secondly, a presentation of files and file type identification is given to provide insight into the problem at hand. Lastly, the technique of clustering is briefly explained to give a basic understanding of its relevance for this thesis.

## 2.1 Digital Forensics

There is an ever-increasing need to store information digitally. The stored information is expected to be available whenever it is needed, and various storage media are depended on to be reliable means of storing it.

Hard disk drives are one of the most popular storage media, and in 2007 it was estimated that over 90% of all digital information is stored on magnetic media such as hard disks [29]. Cloud storage has recently been taken into more use, but even though the data exists in the "cloud", the data is still stored somewhere (e.g.

on hard disks). Unfortunately, this is not as reliable as one would hope for; these magnetic drives are prone to data loss, either in the form of corruption caused by e.g. ageing, or from (un)intentional data deletion.

There are many steps that can be taken to prevent the likelihood of data loss. Data redundancy and geographical distribution of information can mitigate the damage caused by natural disasters, etc., but may not help against malicious users. Properly securing against data loss is very important, both in private households and for companies, but sadly, hard disk drives are relied upon as fail-safe storage mechanisms.

But what if the accident is a reality, and the hard disk with all of the family photographs becomes no longer readable: what can be done? When a hard disk seems unreadable, i.e. the data can not be read through normal means, the assumption that the data is permanently lost can be made prematurely. Depending on the actual cause of data loss, the data is not necessarily lost forever.

### 2.1.1 Files and File type identification

The term *files* is used herein as the reference to a sequence of bytes, representing information stored on a storage medium using an associated file system. The file system manages files and holds information such as the physical location of each file on the medium. This information is stored in a file table.

As previously mentioned, when a file is deleted, modern operating systems just remove the entry of the corresponding file from the file table, keeping the actual data untouched somewhere on the medium. As a file is deleted, the location of the file's data will be marked as available, thus giving the file system permission to store new files in that location.

And when a file is to be written to storage, the file system will try to fill holes of unused space in between other files in order to preserve space. If the file is split

into smaller parts—fragments—that can fit in these holes, the file gets fragmented.

When the storage medium suffers from fragmentation, recovery of a previous file can be difficult because some pieces may be overwritten, and the file's fragments might be scattered throughout the medium. It is therefore recommended that if a file is deleted by accident, no write operations should be performed on the file system until the file has been successfully recovered.

## 2.2 Clustering

Clustering, also known as *Cluster Analysis*, is the process of grouping unlabelled data in groups where selected features are more or less similar. This process separates and groups data, possibly giving structure to unstructured data. Clustering is in itself a theoretical process for which there exists many different implementations for different application areas. A common approach to the clustering of data, is to assess a measurable distance between features, effectively creating distances between data in an arbitrary-dimensional space. For instance, the k-means algorithm uses *euclidian distance* to measure the differences in distance.

# Chapter 3

# Related Work

This chapter presents related work to the proposed solution with respect to the background in the previous chapter.

## 3.1 Digital Forensics

There are many forms of file type identification available both in previous research and publicly available tools [31]. This section briefly describes the extension-, signature- and content-based methods.

The extension based method of file type detection looks at the extension of the actual file name. Operating systems in the Microsoft Windows family use this method extensively to identify the type of a file. It checks the file name for an extension, i.e. the string given after the last period mark.

Using the example file name *image.jpg*, the `jpg` string is the extension, indicating an image file of the JPEG type.

This simple method of type identification is not very robust; hiding the true file type is a simple matter of renaming the file with a different extension. This is

referred to as file spoofing. In other words, the extension alone can not guarantee that a file is of the actual type that it indicates.

As for the signature based approach, file signatures are mainly two bytes that are located at the beginning of binary file types. These signatures are not present in text files, such as `txt`, `html` or `csv` to name a few. *The Sleuth Kit*[10] and *Scalpel*[32] are two of many tools publicly available to identify and recover files based on their signatures.

In UNIX based operating systems such as Linux, these signatures are read in order to determine the type of file the following data belongs to. As with the extension based method, this method can also easily be spoofed. By changing the two signature bytes from a given file, the operating system can be unable to correctly open a file.

Consider the scenario where a UNIX based operating system is to open the file *image.jpg*. Here, the file name is of no indication of the actual type. If this file is actually an executable hiding under the disguise of a JPEG image file, it would not be apparent to the user unless it is explicitly opened in an editor where the signature bytes can be checked. If this file is executed, it will run as an executable because of the signature bytes telling the operating system to run it as such, potentially executing malicious code.

Signature based methods for identifying types of file fragments are also not reliable because with fragmented files, only the first fragment (header) contains the file signature, rendering the remaining fragments without type identification.

File recovery tools often look at the file table when trying to recover files. The file table is an integral part of a file system, and contain information about where on the storage medium the files' data are found. Using this information, files can often be restored as long as the data location has not been overwritten or corrupted.

These tools can also often recover files that have been accidentally deleted;

when a file is deleted, only the data location in the file table is removed from the storage medium, while the actual data of the file still exists somewhere on the medium. Even though the data location is lost through the "deletion" action, file recovery tools can scan for the data and attempt to recover it without using existing information about its location.

An important task when recovering data from storage media is to label the data with some specific type, i.e. to determine what kind of information the data represents. File recovery tools can search for common signatures—specific sequences of bytes—which are often located at the beginning of files, in order to identify the file's type. But if the file has been fragmented, i.e. split into multiple physical pieces on the disk, and the reference to the file is removed from the file table, the file can not so easily be recovered.

It is also important to consider a large amount of files when evaluating methods for file recovery. Quick and Choo [30] investigates some of the challenges related to the increasing volume of data. For a file type identification method to be useful in a real life scenario, it should also be relatively fast since it potentially needs to be run on huge amounts of data.

The content based approach only consider the actual contents of each file in order to determine its type. No operating system is able to easily identify the type of a file based only on its contents unless the contents contain signature bytes at the start of the file.

Common machine learning techniques have been used in order to detect the file type of a fragment. McDaniel and Heydari [26] appear to have been the first in literature to consider the content based approach for file type detection. They consider the use of different classification techniques in order to achieve this. A classification accuracy of up to 96% is reported, i.e 96% of all files were correctly identified. Upon closer inspection, the method yielding their best result is basically utilizing the signature based approach, as it is dependent on the existence of consistent signature bytes within files. This renders their method void in terms of

the content based approach, while their second best, and content based approach, reported a classification accuracy of 46%. A total number of 120 files among 30 file types were considered.

A problem that exists throughout the literature of content based file type identification is the lack of a standardized approach [17]. Results are provided with a varying number of files and types, and they are often produced using private data sets. The validity of the results can therefore not be established through reproduction.

Looking at the work done by Amirani, Toorani, and Beheshti [5], they report a classification accuracy of 85.5% when considering file fragments of 1500 bytes in size, with a total of 1200 files over 6 different file types. Beebe et al. [7] reports a correct classification rate of 73.4% when considering 38 different file types with nearly 100000 fragments of 512 bytes in size. This highlights the problem of not having directly comparable results. One could argue that Beebe et al. [7] presents better results than those of Amirani, Toorani, and Beheshti [5] because of the increased classification accuracy compared to that of a hypothetical random-choice classifier.

On the note of content based file type identification, a large variety of different classifiers are used in literature to solve this problem. Solutions are proposed, many to solve different aspects within file type identification, all with varying published results. Table 3.1 shows many of the most important findings in this field.

Table 3.1: Important works within file type identification (based on table from previous work [15])

| **Contributors** | **File/ Fragment** | **Method** | **#Types** | **#Files** | **Accuracy %** |
|---|---|---|---|---|---|
| McDaniel and Heydari[26] | File | BFA, BFC, FHT analysis | 30 | 120 | 27.5, 45.83, 95.83 |
| Li et al.[25] | File | Manhattan distance<br>Mahalanobis distance<br>Multi-centroid | 8 (5) | 800 | 82 (One-Centroid)<br>89.5 (Multi-Centroid)<br>93.8 (Exempler files) |
| Dunham et al.[13] | File | Neural Networks | 10 | 760 | 91.3 |
| Karresand and Shahmehri[24] | Fragment | Oscar method (based on Mean and standard deviation of BFD).<br>Biased for JPG. | 49 | 53 | 97.9 (JPG) |
| Karresand and Shahmehri[23] | Fragment | Oscar method + rate of change between consecutive byte values.<br>Biased for JPG | 51 | 57 | 87.3-92.1 (JPG)<br>46-84 (ZIP)<br>12.6 (EXE) |
| Zhang et al.[35] | Fragment | BFD and Manhattan distance | 2 | 100 | 92.5 |
| Moody and Erbacher[27] | Fragment | Mean, standard deviation, kurtosis | 8 | 200 | 74.2 |
| Calhoun and Coles[8] | Fragment | Fisher's linear discriminant.<br>Statistical measurements | 2 | 100 | 68.3-88.3 (bytes 129-1024)<br>60.3-86 (bytes 513-1024) |
| Amirani et al.[5] | File | PCA + Neural networks feature extraction.<br>MLP Classifier | 6 | 720 | 98.33 |
| Cao et al.[9] | File | Gram Frequency Distribution, Vector space model | 4 | 1000 | 90.34 (2-gram + 256 grams as type signature |
| Ahmed et al.[1] | File | Cosine similarity, divide and conquer, MLP Classifier | 10 | 2000 | 90.19 |
| Ahmed et al.[3, 2] | Both | Feature Selection, Content Sampling, KNN Classifier | 10 | 5000 | 90.5 (40% of features)<br>88.45 (20% of features) |
| Amirani et al.[6] | Both | PCA + Neural Networks feature extraction<br>SVM Classifier | 6 | 1200 | 99.16 (Whole files)<br>85.5 (1500 bytes fragments)<br>82 (1000 bytes fragments) |
| Evensen et al.[15] (Reference to self) | Both | n-gram analysis with naïve bayes classifier | 6 | 60000 | 99.51 (Whole files)<br>99.08 (8192 bytes fragments, 5 types)<br>98.34 (1024 bytes fragments, 5 types) |

## 3.2 Clustering

Extensive reviews of different clustering algorithms have been performed in literature [34, 4].

There is seemingly no literary work done with respect to file type identification using clustering techniques. This is assumed to be because traditional classification methods have been very successful, foreshadowing the need for additional, alternative approaches.

Clustering generally has many successful areas of application [20, 16], often where traditional classification methods are not applicable. This is especially true for scenarios where there are unknown groups of data, i.e labelling of data cannot be done. A clustering process will then cluster data based on some measurable features to determine equality.

There exists mainly two different types of clustering; hard clustering and fuzzy clustering (also known as soft clustering). In hard clustering, each data point only has one cluster which it belongs to, while for fuzzy clustering, each data point can be members of many clusters, with varying degrees of membership.

For the clustering of text, k-means has proved a rather popular choice. [22, 33, 12, 28, 18, 19, 21]. This is a hard clustering algorithm, assigning each data point to exactly one cluster.

# Chapter 4

# Proposed Solution

The proposed solution presented in this thesis consists of a combination of a classification algorithm and a clustering algorithm. In order to present how these are combined to make up the proposed solution, this chapter first introduces the principles of classification and clustering as it is implemented. The innovation in this thesis is the combination of the supervised classification and unsupervised clustering approaches. Then, the proposed solution is presented, building on principles of both classification and clustering.

An extension of the proposed solution for unsupervised continuous learning is presented, while advantages compared to traditional approaches is lastly discussed.

## 4.1 Basic Classification

Basic supervised learning classifiers utilizes a process of learning. This is done by feeding a classifier features extracted from the data which it is trained with. These features should be class unique, i.e based on the features alone, the class for which the features belong should be identifiable. A predictive model is then generated based on the training data. This model should be capable of labelling new data by comparing its features to that which it has learned. Figure 4.1 illustrates this process.

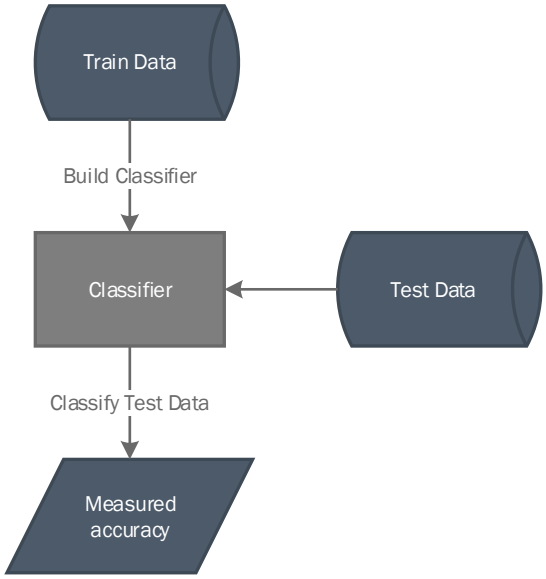


Figure 4.1: Basic classification flow

## 4.2 Basic Clustering

Clustering is an unsupervised learning process which uses measurable features extracted from data to separate them into distinguishable clusters. Depending on the data, different clustering algorithms are suited for different types of data. By clustering unlabelled data, patterns or structures previously unseen can become apparent. Figure 4.2 illustrates the process of clustering.

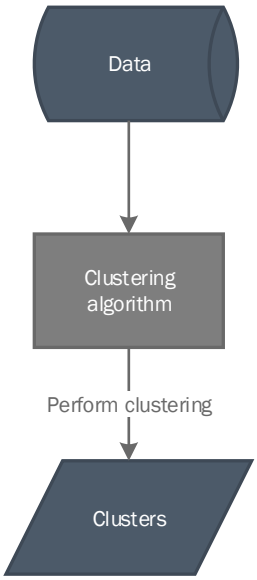


Figure 4.2: Basic clustering flow

## 4.3 Proposed solution

The proposed solution for this thesis utilizes both clustering and classification in order to identify the type of a file, only by looking at its binary contents.

The basic idea is to use labelled test data to build a predictive classification model. The test data is also fed into a clustering algorithm, creating clusters based on the same features which the classifier was trained with. The classifier then uses its knowledge to classify the actual clusters, effectively creating labelled clusters. Some test data, normally unlabelled, are then clustered into the already existing clusters, assigning the data the label of its assigned cluster. This effectively labels unlabelled data in the same way a classifier would, only by clustering the data instead of performing classification of it. The aim is that this method performs faster than that of a traditional classification-only approach. Figure 4.3 illustrates this whole process.

The proposed solution is extended into two additional techniques, *unsupervised continuous learning* and *adaptive learning*, giving it the possibility to learn and adapt to the environment in which it operates.

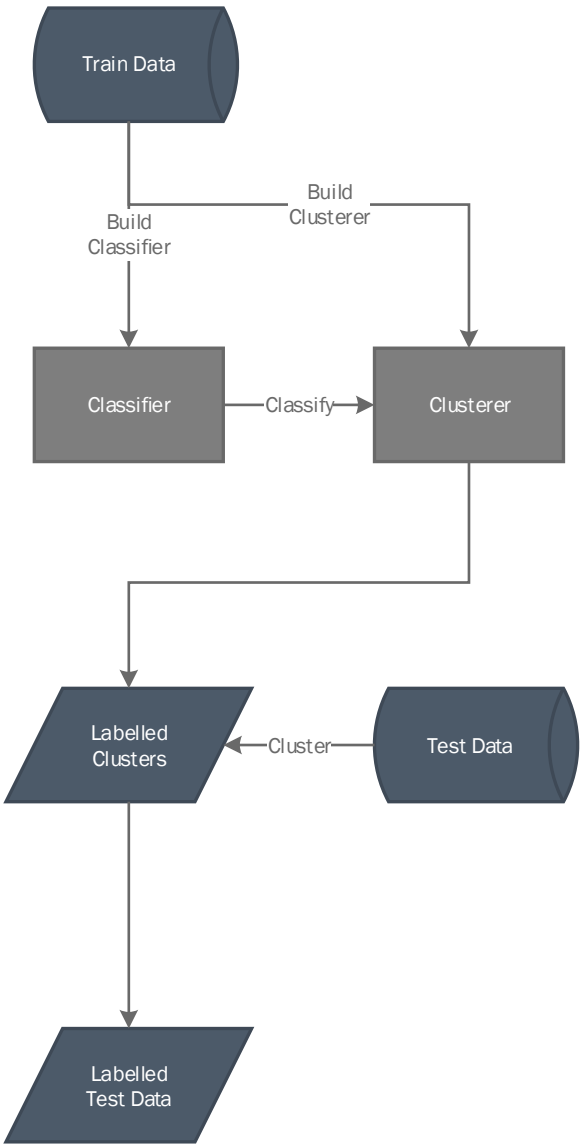


Figure 4.3: Flow of proposed solution

### 4.3.1 Unsupervised Continuous Learning

Expanding on the proposed solution, this thesis investigates how it can be modified so that a method of unsupervised continuous learning can be performed. By splitting some of the test data into smaller parts, the clusters can be built incrementally, effectively learning while clustering. This is performed by first clustering one part of the test data. This part is then fed into the labelled clusters, updating the centroids –the representative for its label– to build a cluster that is assumed to be a better representation of its label due to the continuous learning of test data. Figure 4.4 illustrates this process.

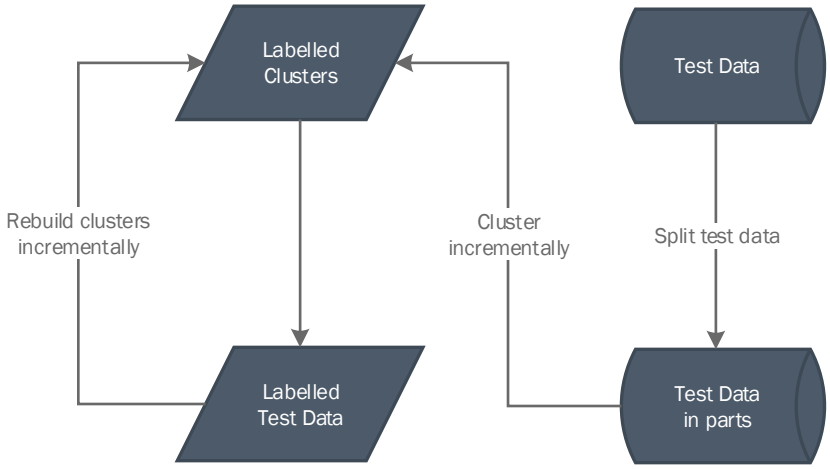


Figure 4.4: Flow of Unsupervised Continuous Learning

### 4.3.2 Adaptive Learning

Further, it is assumed that the technique of unsupervised continuous learning can be extended to include new types of test data, i.e the introduction of new classes. This is herein referred to as *Adaptive Learning*.

To achieve an adaptive learning technique from the proposed solution, new classes of data to the clusterer are introduced, hopefully creating distinct clusters. The centroids of the clusters are then classified by the classifier in order to establish a label for the newly clustered data. This in turn means that the classifier needs to know of all possible classes which may be used as input data for the proposed solution. This may be considered a limitation, but realistically a classifier can be trained with all possible classes of a domain, preparing it for the identification of all possible classes. However, this is assumed to reduce identification accuracy overall. The process of an adaptive learning technique is illustrated with Figure 4.5.

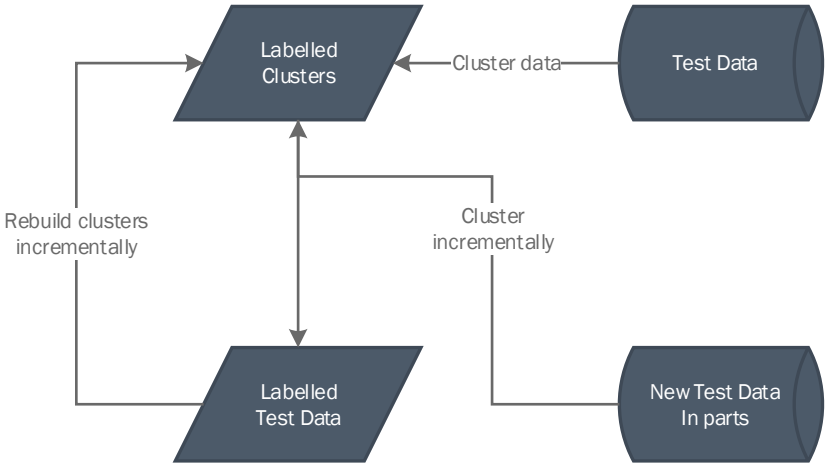


Figure 4.5: Flow of Adaptive Learning

### 4.3.3 Features

A standard approach when considering the identification of file types based on the binary contents of files, is to consider the *Byte Frequency Distribution*(BFD) as features. The BFD is a representation of each byte's occurrence within files, meaning that for each file different bytes are present, making up the whole contents of a file. These bytes range from 0 to 255.

For the BFD to be a viable option when considering features from files, the frequency need to be normalized. This intentionally makes the size of each file irrelevant when considering the frequency of bytes. This is done by dividing the number of each byte with the total number of bytes within each file.

Figures 4.6 through 4.11 depicts the normalized BFDs generated from the data set used in this thesis (See Chapter 5).

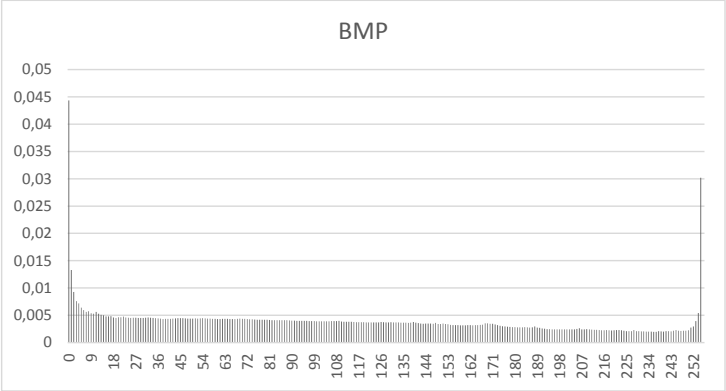


Figure 4.6: BFD for the BMP File format

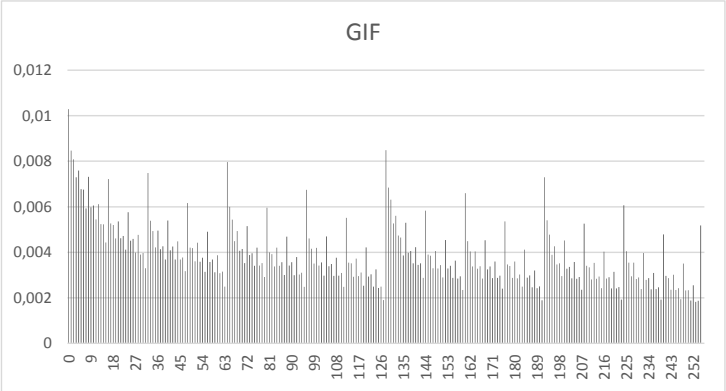


Figure 4.7: BFD for the GIF File format

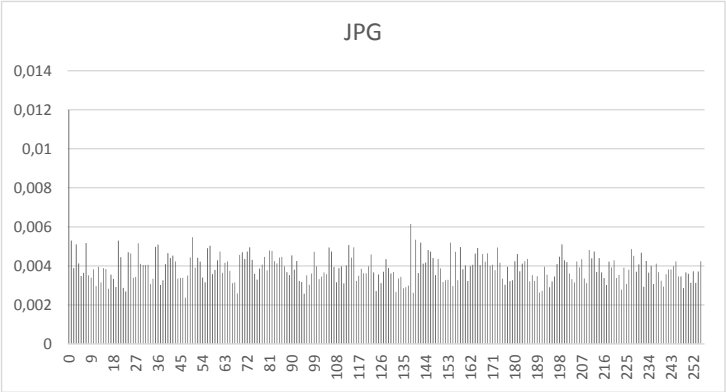


Figure 4.8: BFD for the JPG File format

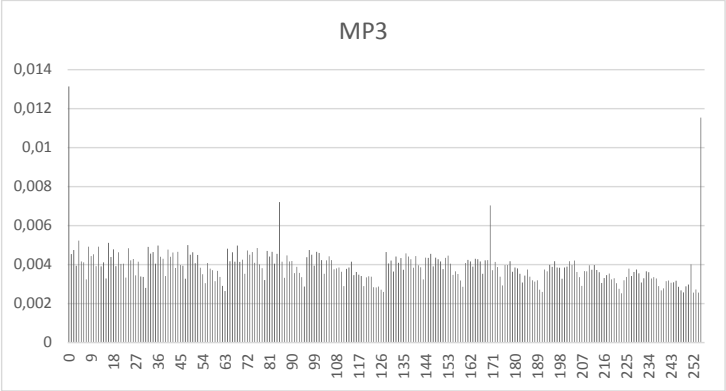


Figure 4.9: BFD for the MP3 File format

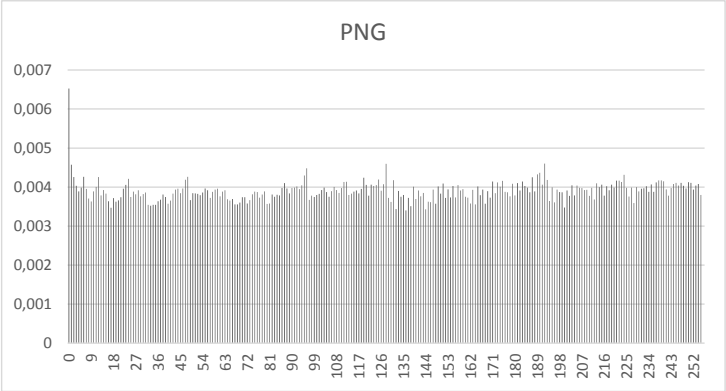


Figure 4.10: BFD for the PNG File format

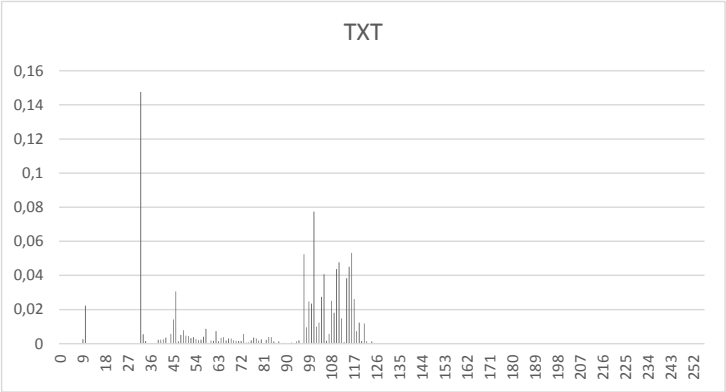


Figure 4.11: BFD for the TXT File format

### 4.3.4 Clustering Algorithm

Previous work [15][1] compared the byte frequency distribution within files to that of letters or words in text classification. This led to the use of the naïve Bayes classifier as the proposed solution with great results. The reason being that the naïve Bayes classifier is a popular choice when dealing with text classification. This assumption is taken further into clustering, i.e since no direct application in previous literature has been found for clustering with file type identification, the principles of text clustering become the neighbouring comparison.

As mentioned in chapter 3, the k-means algorithm has proved a very good approach for the classification of text documents. Hence, k-means is the clustering algorithm of choice in the experiments of this thesis. k-means has proved "extremely fast" [11], reinforcing the decision to consider this algorithm.

### 4.3.5 Advantages

Previous work done in digital forensics, specifically the identification of file types, has mainly been done only with supervised learning techniques. Many of these results have been proven accurate, however some of them operate slowly. Assigning new instances of data to already existing clusters is assumed to be faster than utilizing a traditional supervised learning approach to label the data.

With the introduction of an unsupervised continuous learning technique as well as the adaptive learning technique, it is assumed that the proposed solution is capable of learning while identifying due to its inclusion of clusters which can be easily extended incrementally with new data. This is assumed to increase the rate of which data is correctly identified as it learns and adapts to the environment of the test data, both with regards to already existing classes of test data and newly introduced classes.

[1] Reference to self

# Chapter 5

# Experiments

To achieve a basic implementation of a clustering-classification approach to file type identification, four classification algorithms are compared at first in terms of accuracy; *Support Vector Machine*, *Naïve Bayes Classifier*, *Decision Tree*, and *Multilayer Perceptron*. The best performing classifier from this experiment is considered for the experimentation of the proposed solution. Then, clustering and classification are tested together, where the centroids of clusters created by a k-means algorithm are classified by the forthbrought classifier. This will provide empirical results of the proposed method in this thesis, and hopefully add to its conclusion.

## 5.1 Data set

The data set considered for these experiments consists of six different file types, namely `BMP`, `JPG`, `PNG`, `MP3`, `GIF` and `TXT`. This is also the data set used in previous work [15][1]. The four image types are chosen due to a bias to identify images in digital forensics. This bias is based on serious offences such as the

[1] Reference to self

possession of child pornography. The two additional file types are chosen simply to add two completely different types of data, to prove the possibility of not only image type identification. The files were downloaded as randomly as possible from various sources on the Internet. This data set consists of 10000 files for each type, i.e. a total of 60000 files. For each of the files in the data set, fragments of 512 bytes are extracted. The first 512 bytes are ignored due to the occurrence of file signatures and other file type identifiers in these fragments.

Fragments are chosen instead of whole files, because in a realistic scenario, whole files are often not available, but only fragments of files due to disk fragmentation.

## 5.2 Clustering

The best yielding configuration for the k-means algorithm is found through a series of experiments. Its performance is measured in terms of *purity*. This is a measurement of how pure each cluster is, i.e the sum of the relations of the most populated type assigned to each cluster of the total number of files. This calculation is given by $\sum_{k=1}^{n} \frac{x(k)}{y}$ where $n$ is the total number of clusters, $x$ is a function yielding the number of the most populated type in $k$, and $y$ is the total number of files.

### 5.2.1 k-means: Constant Number of Clusters

In this experiment, 7000 fragments from each file type are clustered. The number of clusters is pre-defined, which is a requirement of the k-means algorithm, with a value six. This is done intuitively because it is the total number of file types in this data set. It is assumed that by forcing the number of clusters equal to the number of types, greater purity is achieved. In a real life scenario the number

of clusters should not be known, but is used here for the sake of measuring the proposed solution.

Table 5.1: k-means clusters with their assigned files

| **Cluster** | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| BMP | 2327 | 1985 | 308 | 65 | 2285 | 30 |
| JPG | 40 | 10 | 0 | 0 | 6950 | 0 |
| PNG | 21 | 14 | 1 | 0 | 6964 | 0 |
| MP3 | 710 | 0 | 5477 | 4 | 804 | 5 |
| GIF | 5 | 6169 | 0 | 0 | 826 | 0 |
| TXT | 1410 | 0 | 0 | 2346 | 26 | 3218 |

Table 5.1 shows the results of this experiment. The five clusters are shown together with the clustered number of each file type.

The clustering purity is measured to 63.1%. The results of this experiment can not be viewed as a good or successful result considering the small number of file types clustered. It is assumed that the reason for the bad performance is because of the small number of possible clusters. Upon closer inspection of the fragments, it is seen that there is variance between files from the same file type, meaning that they may or should be placed in what may be called *sub-classes*. A new experiment is therefore proposed where the number of clusters for the k-means algorithm is increased.

### 5.2.2 k-means: Variable Number of Clusters

For the current data set, the number of clusters is set to increase incrementally from 6, which is the number of file types, to 100. This is an arbitrary large number. Using the given data set, this experiment yields a good performance evaluation with regards to the number of clusters for the k-means algorithm.

Figure 5.1 illustrates the performance in terms of purity, given by the vertical axis, for the k-means algorithm. As seen, and assumed, the purity quickly in-

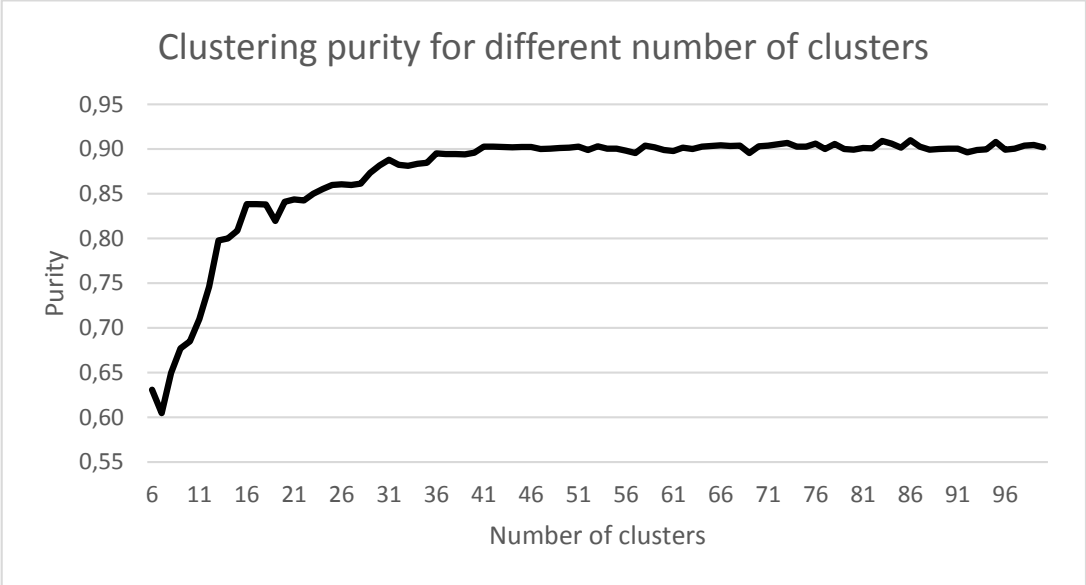


Figure 5.1: k-means performance evaluation: Purity

creases when the number of clusters increases. This is the case for up until about 40 clusters total. From here on out, the purity remains consistent around 90%. However, the elapsed clustering time is assumed to increase when the number of clusters increases. Therefore, a new experiment is proposed where the elapsed clustering time is measured to the numbers of clusters tested for this experiment.

### 5.2.3 k-means: Clustering Time

The same experiment as the previous is performed. This time, instead of measuring purity, the elapsed clustering time for each of the different numbers of clusters are measured. This is depicted in the graph given by Figure 5.2, where the vertical axis indicates the time in seconds. This clearly shows that for a large number of clusters, the clustering time increases seemingly exponentially. Hence, where clustering is time sensitive, the number of clusters selected for the k-means algorithm should be set to a minimal, but sufficient value.

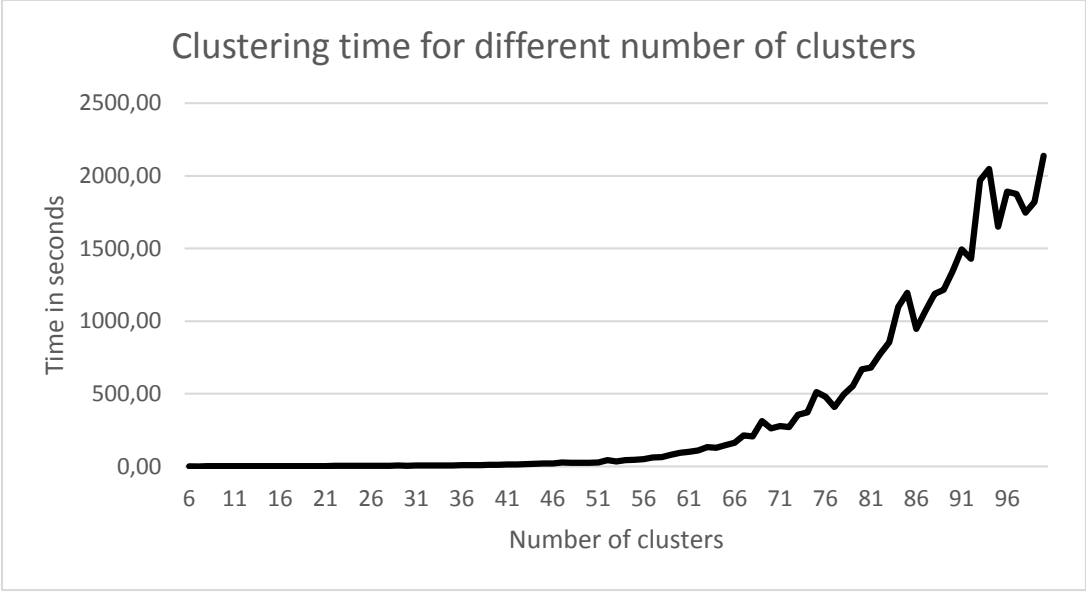


Figure 5.2: k-means performance evaluation: Time

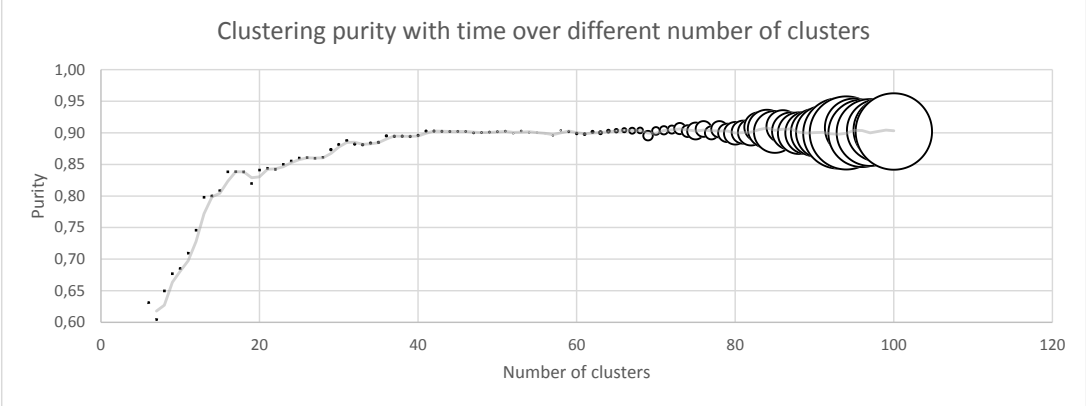


Figure 5.3: k-means performance evaluation: Purity with Time where the bubble size is the relative clustering time

Lastly, to compare purity with regards to time, a bubble chart is presented in Figure 5.3. This depicts the purity in form of bubbles, where the size of each bubble is the relative clustering time for each of the achieved purities.

### 5.2.4 Conclusion

These results clearly indicate that there is indeed a need to increase the number of clusters for the k-means algorithm, to a point well beyond that of the number of assumed classes to cluster. This is assumed to be because of the difference of file contents within each type, effectively creating sub-classes. For the data set considered in these experiments, the achieved purity converges from about 40 clusters and onwards. In terms of clustering time, this is also an acceptable number of clusters. Therefore, the k-means algorithm is considered for the proposed solution with a configuration set to 40 clusters.

## 5.3 Classification

Multiple classification algorithms are measured in terms of the number of correctly assigned fragments from all possible fragments. They are performed on the data set with all 7000 fragments for training and 3000 fragments for classification per file type. For these experiments, the time of learning is not taken into consideration. This due to the principle that the process of training is something to happen once within a domain, while the actual classification happens for every new instance of unknown data, which can be assumed to happen more often than training.

The best performing algorithm for these experiments is chosen as the classifier of choice for the following experiments, evaluating the proposed solution.

### 5.3.1 Support Vector Machine

The Support Vector Machine classifier is considered for experimentation in this thesis due to it being a popular choice within the field of general classification and file type identification.

For this classifier, the *LibSVM* library is used. Its settings are set to that of previous work [14][2], which proved to work well when classifying byte frequency distributions. This means that the SVM classifier is a C-SVC with a kernel type set to *Radial Basis Function*. The parameters are set as following:

**Gamma** 0.0001

**Cost** 8.0

**Epsilon** 0.001

The SVM classifier performed with a classification accuracy of **97.95%**.

[2]Reference to self

### 5.3.2 Naïve Bayes

The naïve Bayes classifier is based on Bayes' Theorem with the naïve assumption of independence between features for a class. Because of this, this classifier is considered simple, but effective is some areas.

The naïve Bayes classifier is the least accurate algorithm with an accuracy of **83.09%**.

### 5.3.3 Decision Tree

Decision trees are based on basic tree structures, where traversal determines the predictive outcome through a series of probabilities given by the branches of the tree.

The Decision Tree implementation in this experiment is done using *J48*, which is an open source Java implementation of the *C4.5* decision tree algorithm. The parameters for this implementation is set as following (based on previous work[14]):

**Confidence** 0.25

**Minimum Instances** 2

**Number of Folds** 3

The classification accuracy is by the Decision Tree given as **94.12%**.

### 5.3.4 Multilayer Perceptron

The Multilayer Perceptron (MLP) algorithm is based on artificial neural networks, which are based on the principles of the biological brain for prediction.

The MLP classifier has its attributes set as following (based on previous work[14]):

- **Learning Rate** 0.3
- **Momentum** 0.2
- **Number of Epochs** 500
- **Error Threshold** 20

Its accuracy is measured to **96.07%**.

### 5.3.5 Conclusion

| Algorithm | Accuracy |
|---|---|
| *Support Vector Machine* | **97.95%** |
| *Naïve Bayes* | **83.09%** |
| *Decision Tree* | **94.12%** |
| *Multilayer Perceptron* | **96.07%** |

As given by these results, the Support Vector Machine classifier clearly outperforms its competition with a classification accuracy of 97.95%. Hence, SVM will be the classifier of choice for the following experiments.

## 5.4 Proposed Solution

An experiment is first performed to establish a comparison between the performance of a traditional classification approach to that of the proposed solution. This is done by measuring both time and classification accuracy on multiple sizes of data sets. This yields a performance evaluation in terms of speed and accuracy, as well as scalability. Secondly, an experiment is performed to examine the performance of the unsupervised learning technique. Lastly, the possibility of introducing a new file type to the unsupervised learning technique is established and presented in the last experiment.

### 5.4.1 Experiment 1 - Speed, Accuracy and Scalability

The SVM classifier is trained with 7000 fragments from the data set. Then, classification is performed on data sets of 30, 150, 300, 750, 1500 and 3000 fragments.

The results yielded by the classifier are given by Table 5.2.

Table 5.2: Experiment 1 - Classification Accuracy and Time

| # Test files | Correctly assigned files | Time in seconds |
|---|---|---|
| 30 | 97.22% | 0.93 |
| 150 | 98.78% | 4.17 |
| 300 | 98.78% | 8.35 |
| 750 | 98.58% | 20.74 |
| 1500 | 98.47% | 41.42 |
| 3000 | 97.95% | 83.47 |

As seen, the classification accuracies are consistently high no matter how many fragments are classified. This is expected due to the classifier being trained with 7000 fragments through all classifications.

Next, the proposed solution is measured on the same test data which was classified, also measuring accuracy and speed. These results are given by Table 5.3.

Table 5.3: Experiment 1 - Proposed Solution Accuracy and Time

| # Test files | Correctly assigned files | Time in seconds |
|---|---|---|
| 30 | 73.33% | 0.02 |
| 150 | 85.78% | 0.10 |
| 300 | 86.78% | 0.21 |
| 750 | 87.02% | 0.52 |
| 1500 | 87.03% | 1.03 |
| 3000 | 85.97% | 2.06 |

Although the accuracy of the proposed solution is lower in all cases compared to that of only classification, the speed of which the fragments are labelled is significantly increased.

These results provide evidence towards the assumption that clustering is faster than classification, at least for the case of a k-means clustering algorithm and an SVM classifier.

Hence, for the current clustering and classification algorithms, the first research question has been answered; the proposed solution is faster than traditional classification approaches. However, since the accuracy of the proposed solution is lower than that of only a classifier, the practical usage of the proposed solution is very situational. In a scenario where there is emphasis on speed, the proposed solution would be considered a better choice compared to an SVM classifier. An example of a scenario with emphasis on speed is the monitoring of network traffic where it is important to read and label every package as they pass by.

### 5.4.2 Experiment 2 - Unsupervised Continuous Learning

Expanding the proposed solution, an experiment is performed to assess the possibility of an unsupervised continuous learning technique.

For this experiment, the test data is split into smaller sets of 1000 fragments. The order of the fragments are first randomized to eliminate the possibility of contiguity of file types. This is done with 3000 fragments per type, yielding a total of 18000 fragments, i.e 18 subsets of 1000 fragments are created.

To establish a baseline comparison, all of these subsets are first normally identified using the proposed solution. The identification accuracies are given by Table 5.4.

Table 5.4: Normal identification of subsets

| Iteration | Accuracy |
|---|---|
| 0 | 86,6 % |
| 1 | 84,5 % |
| 2 | 86,5 % |
| 3 | 85,8 % |
| 4 | 86,5 % |
| 5 | 85,3 % |
| 6 | 86,7 % |
| 7 | 84,8 % |
| 8 | 84,5 % |
| 9 | 86,9 % |
| 10 | 85,3 % |
| 11 | 85,8 % |
| 12 | 86,8 % |
| 13 | 86,8 % |
| 14 | 87,7 % |
| 15 | 86,2 % |
| 16 | 85,1 % |
| 17 | 85,6 % |

As seen, these results are expectedly similar to those of Experiment 1.

To achieve unsupervised continuous learning, these subsets are added to the existing clusters as they are identified. This is done by iteratively recalculating the centroids of each cluster with the new data, purposely creating a more accurate representation of each labelled cluster. This is done utilizing the process described in subsection 4.3.1. The results achieved from this process is given by Table 5.5.

Table 5.5: Identification of subsets by Unsupervised Continuous Learning

| Iteration | Accuracy |
|---|---|
| 0 | 86,6 % |
| 1 | 88,2 % |
| 2 | 88,3 % |
| 3 | 90,4 % |
| 4 | 89,1 % |
| 5 | 87,6 % |
| 6 | 87,9 % |
| 7 | 88,8 % |
| 8 | 87,2 % |
| 9 | 89,9 % |
| 10 | 88,7 % |
| 11 | 87,8 % |
| 12 | 89,5 % |
| 13 | 88,7 % |
| 14 | 90,4 % |
| 15 | 90,2 % |
| 16 | 87,3 % |
| 17 | 87,9 % |

The method of unsupervised continuous learning clearly achieves better results than that of normal identification of each of the subsets. This is also portrayed in Figure 5.4 as a comparison. Purity is also included to show the maximum attainable identification accuracy. Purity remains more or less constant between the comparisons. One important thing to note here, is that identification accuracy will never increase beyond that of the purity of the clustering algorithm. This is because purity is a measurement of the occurrence of the most populated class

within a cluster. This in turn means that if a cluster is labelled equal to its most populated class, each point of that class is correctly labelled, yielding the highest possible purity.

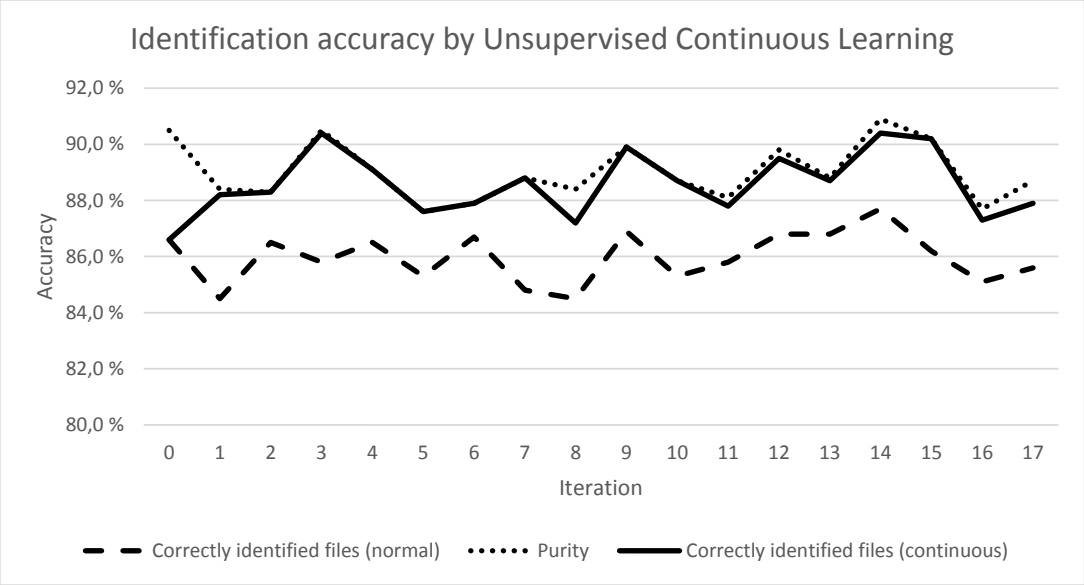


Figure 5.4: Experiment 2 - Unsupervised Continuous Learning

As seen by this experiment, introducing a method of unsupervised continuous learning increases the identification accuracy. However, the purity of the clusters are a limiting factor to the highest achievable identification accuracy. This means that if another clustering algorithm is capable of achieving a higher purity based on the features considered in this thesis, the overall results may increase beyond that which is shown here. Hence, the possibility of an unsupervised continuous learning technique is theoretically shown here, but the consideration of other clustering algorithms are out of scope from this thesis.

### 5.4.3 Experiment 3 - Adaptive Learning

Extending the technique of unsupervised continuous learning, this experiment investigates the possibility of introducing a new class to the clusters by continuously building the clusters.

To test this technique, a new file type is introduced; `PDF`. A data set of 3000 fragments are gathered and created, then split into subsets of 100 files, yielding a total of 30 subsets.

The classifier is learned using the six previously mentioned file types, as well as a separate dataset containing PDF files. The clusters contain only knowledge of the six original file types, as is done in Experiment 1.

The subsets containing PDF files are then clustered iteratively while continuously learning with each subset, i.e the centroids of the clusters are again classified, labelling them. Figure 5.5 portrays the results achieved from this experiment.

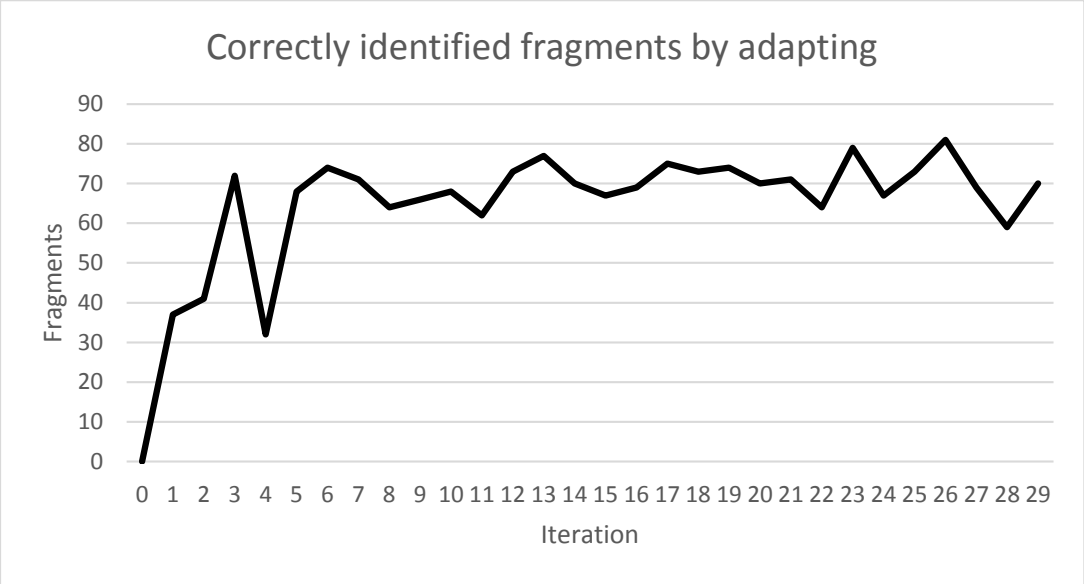


Figure 5.5: Experiment 3 - Adaptive Learning

As seen, none of the fragments from the first subset is correctly identified. This is because there exist no labelled cluster for the PDF file type. For the next subset, some of the fragments are indeed identified. This is because the classifier now has labelled PDF-containing clusters as PDF.

The results vary greatly when few fragments have been clustered, while stability is achieved the more fragments are clustered. This is because a more representative model of the PDF class is generated with more fragments.

Although the results are not great, it has been theoretically shown that the proposed solution is capable of adapting to new file types. This is of course limited to the fact that the classifier needs to know of the file types to be able to label new clusters, but this may be the case in a realistic scenario.

### 5.4.4 Discussion of Results

The first experiment investigated the performance of the proposed solution for file type identification. The results provided evidence toward it being faster, though less accurate, than traditional supervised learning approaches. This means that in cases where speed is of the essence, and accuracy is of less importance, the proposed solution is a viable choice for file type identification in digital forensics.

The second experiment extended the proposed solution to a technique of unsupervised continuous learning, where the proposed solution continuously learns from data. This increased the identification accuracy slightly with little impact on speed, due to the addition of points to the clusters being a non-costly operation.

Lastly, the third experiment extended the unsupervised continuous learning technique to an adaptive learning technique that continuously learns and adapts to the environment of which it operates. This experiment was performed by introducing a new file type to the proposed solution to see whether or not it was capable of adapting to a completely new type of data. The results gave positive

results, meaning that it was in fact able to learn and identify the new type of data, effectively proving its adaptability.

These results provide evidence towards the proposed solution being a viable choice for file type identification in digital forensics. Although the identification accuracy is below that of traditional supervised learning techniques, the proposed solution provides advantages such as speed, continuous learning and adaptability to its environment.

# Chapter 6

# Conclusion and further work

This chapter presents the summary of this thesis, discusses and concludes the results, and presents possible further work.

## 6.1 Summary of Results

The experimentation of the proposed solution presented in this thesis started by investigating suitable clustering and classification algorithms. This resulted in the choice to use a k-means clustering algorithm. This was because of the aforementioned similarity between that of text and binary representations. As a classification algorithm, an SVM classifier was chosen based on its identification accuracy compared to that of other relevant algorithms. As features for these algorithms, the byte frequency distributions of file fragments were chosen.

The first experiment utilized these algorithms to test the proposed solution, establishing its performance in terms of speed, accuracy and scalability compared to traditional classification approaches. The proposed solution achieved a greater speed that traditional approaches. However, this impacted the total identification

accuracy. When classification was performed using an SVM classifier on 3000 files, the elapsed classification time was 83.47 seconds and its identification accuracy was 97.95%. The proposed solution had a running time of 2.06 seconds, but its identification accuracy was 85.97%, yielding less accurate results at a faster speed. The practical usage for this is therefore considered situational. In cases where speed is more important than near-perfect identification accuracy, the proposed solution may prove a valuable choice. As for the scalability, it is seen that both methods scale linearly in terms of time when the number of files increases.

The second experiment investigated how the proposed solution performed with the proposed technique of unsupervised continuous learning. The test data set used for Experiment 1 were split into smaller subsets, containing 1000 files each. This resulted in 18 iterations where the proposed solution continuously learned based on previous iterations. Compared to separate identifications of the subsets, the continuous learning technique slightly increased the identification accuracy as it learned. The most significant increase happened at iteration 4, where standard identification resulted in an identification accuracy of 85.8%, while by reinforcing, an identification accuracy of 90.4% was achieved.

The third and last experiment established the possibility of adaptively learning to identify new file types as they are introduced. 3000 fragments of a new file type was introduced and split into subsets of 100 files each, creating 30 subsets. 30 iterations of this experiment was therefore performed, learning through each iteration. When identifying and learning through the subsets, the identification accuracy gradually increased, stabilizing around 70% identification accuracy. This experiment proved in theory that the proposed solution is capable of adapting to new file types. However, this is limited to the fact that the classifier need to be learned with the new file types in order to be able to classify the centroids of the clusters these new fragments are inserted.

## 6.2 Conclusion

The work in this thesis started by aiming to improve on the speed of file type identification within digital forensics. The motivation for doing this was that digital forensics teams are often presented with huge amounts of data, resulting in a very long identification time. This may not always be practically feasible, as evidence may need to be found within a limited time frame.

As seen with the first experiment, an increase in speed of file type identification was indeed achieved, although this resulted in a decrease of total identification accuracy. These results are limited to the considered algorithms, both for clustering and classification, within this thesis.

The proposed solution was extended to a method of unsupervised continuous learning. This resulted in it successfully continuously learning while identifying, both with test data and the introduction of a new file type. These experiments provide evidence towards the proposed solution being able to adapt the environment of which it operates. This is however limited to the fact that the classifier needs to know of all possible file types to be able to label the clusters to that of which the new data is assigned.

This concludes the proposed solution and this thesis successfully. The research questions were indeed answered, and the results provide evidence toward it being a viable choice in some cases of file type identification within the field of digital forensics.

## 6.3 Further Work

As previously mentioned, the results of this thesis are limited to the chosen algorithms for clustering and classification. A different choice of algorithms may result in different results. It is therefore proposed as further work to experiment

with different combinations of clustering and classification algorithms to potentially increase the overall results achieved herein.

# Bibliography


[1] Irfan Ahmed et al. “Content-based File-type Identification Using Cosine Similarity and a Divide-and-Conquer Approach”. In: *IETE Technical Review* 27.6 (Nov. 2010), pp. 465–477. ISSN: 0256-4602. DOI: 10.4103/02564602.2010.10876780. URL: http://www.tandfonline.com/doi/abs/10.4103/02564602.2010.10876780 (visited on 03/17/2015).

[2] Irfan Ahmed et al. “Fast Content-Based File Type Identification”. en. In: *Advances in Digital Forensics VII*. Ed. by Gilbert Peterson and Sujeet Shenoi. IFIP Advances in Information and Communication Technology 361. Springer Berlin Heidelberg, Jan. 2011, pp. 65–75. ISBN: 978-3-642-24211-3, 978-3-642-24212-0. URL: http://link.springer.com/chapter/10.1007/978-3-642-24212-0_5 (visited on 10/27/2014).

[3] Irfan Ahmed et al. “Fast File-type Identification”. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC ’10. New York, NY, USA: ACM, 2010, pp. 1601–1602. ISBN: 978-1-60558-639-7. DOI: 10.1145/1774088.1774431. URL: http://doi.acm.org/10.1145/1774088.1774431 (visited on 10/27/2014).

[4] Ramiz M. Aliguliyev. “Performance evaluation of density-based clustering methods”. In: *Information Sciences* 179.20 (Sept. 2009), pp. 3583–3602. ISSN: 0020-0255. DOI: 10.1016/j.ins.2009.06.012. URL:

http://www.sciencedirect.com/science/article/pii/S0020025509002564 (visited on 02/26/2015).

[5] M.C. Amirani, M. Toorani, and A. Beheshti. "A new approach to content-based file type detection". In: *IEEE Symposium on Computers and Communications, 2008. ISCC 2008*. July 2008, pp. 1103–1108. DOI: 10.1109/ISCC.2008.4625611.

[6] Mehdi Chehel Amirani, Mohsen Toorani, and Sara Mihandoost. "Feature-based Type Identification of File Fragments". en. In: *Security and Communication Networks* 6.1 (Jan. 2013), pp. 115–128. ISSN: 1939-0122. DOI: 10.1002/sec.553. URL: http://onlinelibrary.wiley.com/doi/10.1002/sec.553/abstract (visited on 10/24/2014).

[7] N.L. Beebe et al. "Sceadan: Using Concatenated N-Gram Vectors for Improved File and Data Type Classification". In: *IEEE Transactions on Information Forensics and Security* 8.9 (Sept. 2013), pp. 1519–1530. ISSN: 1556-6013. DOI: 10.1109/TIFS.2013.2274728.

[8] William C. Calhoun and Drue Coles. "Predicting the types of file fragments". In: *Digital Investigation*. The Proceedings of the Eighth Annual DFRWS Conference 5, Supplement (Sept. 2008), S14–S20. ISSN: 1742-2876. DOI: 10.1016/j.diin.2008.05.005. URL: http://www.sciencedirect.com/science/article/pii/S1742287608000273 (visited on 10/24/2014).

[9] Ding Cao et al. "Feature selection based file type identification algorithm". In: *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*. Vol. 3. Oct. 2010, pp. 58–62. DOI: 10.1109/ICICISYS.2010.5658559.

[10] Brian Carrier. *The Sleuth Kit (TSK) & Autopsy: Open Source Digital Forensics Tools*. URL: http://www.sleuthkit.org/ (visited on 11/05/2014).

[11] Adam Coates, Andrew Y. Ng, and Honglak Lee. “An analysis of single-layer networks in unsupervised feature learning”. In: *International Conference on Artificial Intelligence and Statistics*. 2011, pp. 215–223. URL: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2011_CoatesNL11.pdf (visited on 03/20/2015).

[12] Inderjit S. Dhillon, James Fan, and Yuqiang Guan. “Efficient Clustering of Very Large Document Collections”. en. In: *Data Mining for Scientific and Engineering Applications*. Ed. by Robert L. Grossman et al. Massive Computing 2. Springer US, 2001, pp. 357–381. ISBN: 978-1-4020-0114-7, 978-1-4615-1733-7. URL: http://link.springer.com/chapter/10.1007/978-1-4615-1733-7_20 (visited on 03/16/2015).

[13] J.G. Dunham, Ming-Tan Sun, and J.C.R. Tseng. “Classifying file type of stream ciphers in depth using neural networks”. In: *The 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005*. 2005, pp. 97–. DOI: 10.1109/AICCSA.2005.1387088.

[14] John Daniel Evensen and Stian Guttormsen. “File Type Identification: A Standardized Approach”.

[15] John Daniel Evensen, Sindre Lindahl, and Morten Goodwin. “File-type Detection Using Naïve Bayes and n-gram Analysis”. In: *Norsk informasjonssikkerhetskonferanse (NISK)* 7.1 (2014). URL: http://ojs.bibsys.no/index.php/NISK/article/view/54 (visited on 12/11/2014).

[16] B. S. Everitt, S. Landau, and M. Leese. “Cluster Analysis Arnold”. In: *A member of the Hodder Headline Group, London* (2001).

[17] Simson L. Garfinkel. “Digital forensics research: The next 10 years”. In: *Digital Investigation*. The Proceedings of the Tenth Annual DFRWS Conference 7, Supplement (Aug. 2010), S64–S73. ISSN: 1742-2876. DOI: 10.1016/j.diin.2010.05.009. URL: http://www.sciencedirect.com/science/article/pii/S1742287610000368 (visited on 10/27/2014).

[18] Johannes Grabmeier and Andreas Rudolph. “Techniques of Cluster Algorithms in Data Mining”. en. In: *Data Mining and Knowledge Discovery* 6.4 (Oct. 2002), pp. 303–360. ISSN: 1384-5810, 1573-756X. DOI: 10.1023/A:1016308404627. URL: http://link.springer.com/article/10.1023/A:1016308404627 (visited on 03/16/2015).

[19] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining, Southeast Asia Edition: Concepts and Techniques*. en. Morgan Kaufmann, Apr. 2006. ISBN: 9780080475585.

[20] John A. Hartigan. “Clustering algorithms”. In: (1975). URL: http://cds.cern.ch/record/105051 (visited on 03/16/2015).

[21] A. K. Jain, M. N. Murty, and P. J. Flynn. “Data Clustering: A Review”. In: *ACM Comput. Surv.* 31.3 (Sept. 1999), pp. 264–323. ISSN: 0360-0300. DOI: 10.1145/331499.331504. URL: http://doi.acm.org/10.1145/331499.331504 (visited on 03/16/2015).

[22] Liping Jing et al. “Subspace Clustering of Text Documents with Feature Weighting K-Means Algorithm”. en. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Tu Bao Ho, David Cheung, and Huan Liu. Lecture Notes in Computer Science 3518. Springer Berlin Heidelberg, 2005, pp. 802–812. ISBN: 978-3-540-26076-9, 978-3-540-31935-1. URL: http://link.springer.com/chapter/10.1007/11430919_94 (visited on 03/16/2015).

[23] M. Karresand and N. Shahmehri. “File Type Identification of Data Fragments by Their Binary Structure”. In: *2006 IEEE Information Assurance Workshop*. June 2006, pp. 140–147. DOI: 10.1109/IAW.2006.1652088.

[24] Martin Karresand and Nahid Shahmehri. “Oscar — File Type Identification of Binary Data in Disk Clusters and RAM Pages”. en. In: *Security and Privacy in Dynamic Environments*. Ed. by Simone Fischer-Hübner et al. IFIP International Federation for Information Processing 201. Springer US, Jan. 2006, pp. 413–424. ISBN: 978-0-387-33405-9, 978-0-387-33406-6. URL:

http://link.springer.com/chapter/10.1007/0-387-33406-8_35 (visited on 10/27/2014).

[25] Wei-Jen Li et al. "Fileprints: identifying file types by n-gram analysis". In: *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*. June 2005, pp. 64–71. DOI: 10.1109/IAW.2005.1495935.

[26] M. McDaniel and M.H. Heydari. "Content based file type detection algorithms". In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003*. Jan. 2003, pages. DOI: 10.1109/HICSS.2003.1174905.

[27] S.J. Moody and R.F. Erbacher. "SADI - Statistical Analysis for Data Type Identification". In: *Third International Workshop on Systematic Approaches to Digital Forensic Engineering, 2008. SADFE '08*. May 2008, pp. 41–54. DOI: 10.1109/SADFE.2008.13.

[28] J. M Peña, J. A Lozano, and P Larrañaga. "An empirical comparison of four initialization methods for the K-Means algorithm". In: *Pattern Recognition Letters* 20.10 (Oct. 1999), pp. 1027–1040. ISSN: 0167-8655. DOI: 10.1016/S0167-8655(99)00069-0. URL: http://www.sciencedirect.com/science/article/pii/S0167865599000690 (visited on 02/12/2015).

[29] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. "Failure Trends in a Large Disk Drive Population." In: *FAST*. Vol. 7. 2007, pp. 17–23. URL: http://static.usenix.org/event/fast07/tech/full_papers/pinheiro/pinheiro_html/ (visited on 11/19/2014).

[30] Darren Quick and Kim-Kwang Raymond Choo. "Impacts of increasing volume of digital forensic data: A survey and future research challenges". In: *Digital Investigation* (Dec. 2014). ISSN: 1742-2876. DOI: 10.1016/j.diin.2014.09.002. URL: http://www.sciencedirect.

com/science/article/pii/S1742287614001066 (visited on 11/03/2014).

[31] Sriram Raghavan. "Digital forensic research: current state of the art". en. In: *CSI Transactions on ICT* 1.1 (Mar. 2013), pp. 91–114. ISSN: 2277-9078, 2277-9086. DOI: 10.1007/s40012-012-0008-7. URL: http://link.springer.com/article/10.1007/s40012-012-0008-7 (visited on 11/03/2014).

[32] Golden G. Richard III and Vassil Roussev. "Scalpel: A Frugal, High Performance File Carver". In: *DFRWS* (2005). URL: http://www.dfrws.org/2005/proceedings/richard_scalpel.pdf (visited on 10/24/2014).

[33] M. Steinbach, G. Karypis, and V. Kumar. "A Comparison of Document Clustering Techniques". In: (). URL: http://www.cs.sfu.ca/~wangk/894report/chen1.pdf (visited on 03/16/2015).

[34] Rui Xu and II Wunsch D. "Survey of clustering algorithms". In: *IEEE Transactions on Neural Networks* 16.3 (May 2005), pp. 645–678. ISSN: 1045-9227. DOI: 10.1109/TNN.2005.845141.

[35] Like Zhang and Gregory B. White. "An Approach to Detect Executable Content for Anomaly Based Network Intrusion Detection." In: *IPDPS*. 2007, pp. 1–8. URL: http://www.researchgate.net/publication/220952472_An_Approach_to_Detect_Executable_Content_for_Anomaly_Based_Network_Intrusion_Detection/file/e0b4952220bc77f49f.pdf (visited on 10/27/2014).