Forensic Science International: Digital Investigation 36 (2021) 301121

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

Full Paper

A comparative study of support vector machine and neural networks for file type identification using n-gram analysis

Joachim Sester [a], Darren Hayes [b], Mark Scanlon [c], Nhien-An Le-Khac [c, *]

[a] Bundesministrium des Inneren (BMI), NRW, Germany
[b] Pace University, New York, NY, USA
[c] University College Dublin, Belfield, Dublin 4, Ireland

ARTICLE INFO

Article history:
Available online 23 March 2021

Keywords:
File type identification
n-grams analysis
Forensic analysis
Neural networks
Support vector machine

ABSTRACT

File type identification (FTI) has become a major discipline for anti-virus developers, firewall designers and for forensic cybercrime investigators. Over the past few years, research has seen the introduction of several classifiers and features. One of these advances is the so-called n-grams analysis, which is an interpretation of statistical counting in classified fragments. Recently, n-grams based approaches were already successfully combined with computational intelligence classifiers. However, the academic body of literature is scant when it comes to a comprehensive explanation of machine learning based approaches such as neural networks (NN) or support vector machines (SVM). For example, how the input parameters, including learning rate, different values of n for n-grams, etc. influence the results. In addition, very few studies have compared the scalability of NN vs. SVM approaches. Therefore, a systematic research in comparing different approaches is needed to address these questions. Hence, this paper investigates this type of comparison, by focusing on the n-gram analysis as a feature for the two different classifiers: SVMs and NNs. This paper details our experiments with two NNs and four SVMs, using linear kernels and RBF kernels on RealDC datasets. In general, we found that SVM-based approaches performed better than the NN, but their scalability is still a challenge.

1. Introduction

Although data recognition in the past was primarily intended to assist a user, nowadays it is used in many products, such as intelligent firewalls or forensic diagnostic applications. Additionally, successfully recognizing spoofed files is becoming increasingly important. In some cases, the evaluation of a single data source in an investigation can take years. Thus, methods have been developed that use artificial intelligence, which is capable of finding files and classifying their origin type. This could refer to deleted volumes, hidden data in other files or within data streams, perhaps deliberately misnamed file extensions, disk or block level slack space and unallocated space (Du et al., 2018). Research in this area is increasingly focused on the correct identification of files, especially the file type recognition. Today, this research is an integral part of the technology in our society. Many attack scenarios are based on the new types of intrusion data. Therefore, effective identification is essential. Thus, researching the file type identification (FTI) is not only about recognizing files but it is also about the real-time monitoring systems that can detect, recognize and prevent attacks on businesses and states.

Although the topic of FTI has been widely studied, several researchers have proposed new methods without testing them against all of the previously existing methods. Hence, some problems have now arisen that require a detailed reappraisal of previous approaches. The training and testing of neural networks (NN), for FTI on public data corpora, seems to be a gap in previous scientific work. Furthermore, the usage of Support Vector Machines (SVMs) (Vapnik, 1998) with different kernels was not exhaustively performed, as there is an increasing number of possibilities. Li et al. (2010) did work with monograms on SVM with several kernels, and Gopal et al. (2011) used SVMs and k-nearest neighbour (kNN) to compare the performance with mono- and bigrams, but the direct comparison between SVMs and NNs on public data sets has not been performed. Therefore, in this paper we aim to perform systematic research on the performance of machine learning-based approaches of FTI by studying the influence of different setups on input parameters, such as a different *n* for *n-gram* analysis to

* Corresponding author.
E-mail address: an.lekhac@ucd.ie (N.-A. Le-Khac).

https://doi.org/10.1016/j.fsidi.2021.301121

classification results. We also discuss the scalability of NNs approach vs. SVMs. The main contributions of this paper can be listed as follows:

- A comprehensive survey on FTI approaches in the context of cyber security and forensics in the literature.
- A comparative study of NNs and SVM approaches for FTI, by using n-gram analysis.
- Intensive experiments on different NNs and SVM algorithms with a popular dataset.

2. File type identification: survey

The topic of FTI has been studied for years (Roussev and Garfinkel, 2009). Several taxonomies were developed and researched. The first split in topics was the differentiation between identifying files or fragments of files. Garfinkel (2010) stated that there are no "common schemas, file formats, and ontologies". Rainer Poisel et al. (2014) categorized the whole research area into five main classes: signature-based approaches, statistical approaches, computational intelligence-based approaches, approaches considering the context, and other approaches (Poisel et al., 2014). This section focuses only on the three most popular classes: signature-based, statistical and computational intelligence-based approaches.

2.1. Signature-based approaches

The first category describes techniques based on a comparison of known to unknown file fragments. Probably the best-known method is the so-called "magic byte file" in UNIX operating systems. The "file" command extracts several bytes located at the beginning or end of the examined file and compares it with a list of pre-defined values. Many popular carving tools use this method for matching file types. This method was extended by Pal et al. (Pal and Memon, 2009; Nguyen Thi et al., 2017) for the use of header- and footer byte-sequences for "syntactical tests". Garfinkel et al. (2010) also worked on this approach by adding further signatures. Then, hashing emerged and was discussed by Ruback et al. (2012), who used data mining techniques to create hash sets. The hashing of files did not help with analysing unknown file types but rather helps to find well-known files. This method of hashing was further researched, by authors like Chawathe (2009), who used the locality-sensitive hashing scheme to cluster similar file types (Gionis et al., 1999). Garfinkel (2006) and Dandass et al. (2008) proposed the use of hash-values for fragments to identify uniquely files with the same fragments. They also recommended a change of hashing algorithms, including MD5 and SHA1 to CRC32, as it is shorter and has a "comparably low false-positive rate" (Poisel et al., 2014). A further speed enhancement was provided by Collange et al. (2009), who proposed using Graphical Processing Units (GPUs), such as "hashcat", plus a selective hashing of signatures for the disk sector, also known as "sector hashing" (Garfinkel and McCarrin, 2015). The problem was that these hashes only worked on fixed, predefined block sizes. Streams of data and single, especially small files were not affected. Garfinkel et al. (2010) contributed further by researching a faster way to match master files with image files by using maps. Bloom filters were then introduced (Garfinkel et al., 2010; Young et al., 2012) with similarity preserving hashing (SPH) (Roussev et al., 2010), an entropy-based scheme for feature selection and another length compression algorithm (Roussev, 2011) advanced processing speed. Finally, the tool "sdhash" was introduced by Roussev (Roussev et al., 2010), which performed with 94% accuracy, and was superior to "ssdeep", with 68% accuracy and was developed by Kornblum (2006). Consequently, Roussev and Quates (2012) started thinking about "sdhash" real-time digital forensics and triage, and Roussev et al. (2013) estimated a necessary calculation speed of 120 MB/s, which could be achieved with 120–200 computing cores. Continuing these efforts, Breitinger et al. (2013) advanced the research with similarity preserving hashing (SPH). Subsequently, expediting the lookup speed through hierarchical Bloom filter tress was achieved by Lillis et al. (2018).

2.2. Statistical approaches

The second class of approaches, called the "Statistical Approaches", combines those techniques that make use of the characteristic attributes of the file's content. Typically, counting bytes and the interpretation of those in comparison to other bytes are used. The usual representatives of this kind are informational entropy by Claude Shannon (1948) and binary frequency distribution (BFD) by McDaniel and Heydari (2003). Table 1 provides an overview of the progression. This table, referred to by many previous works (Daniel Evensen, 2015; Amirani et al., 2013), is a well-tended classic overview that also includes research beyond fragmentation-FTI.

McDaniel and Heydari (2003) published a paper introducing three new methods of content-based FTI: Byte frequency analysis (BFA), The Byte Frequency Cross-Correlation (BFC) and the File Header/Trailer (FHT) algorithm. The BFA algorithm determines the number of occurrences of bytes (8 Bits <=> 256-byte values). This sort of pre-processing is referenced as Byte Frequency Distribution (BFD). It is also known as monogram statistic, 1-g or byte histogram. These values, independent of a file's size, are divided by the total number of bytes, normalizing them to a scale between zero and one. This BFD indicates different characteristic patterns, based upon the file's type. During the testing, the BFA algorithm classified 27.5% of 30 different file types correctly.

The second method, Byte Frequency Cross-Correlation (BFC) does not just consider the relative difference, but also the absolute byte's values. This extension of the BFA has an accuracy of 45.83%. The weakness, however, was the incomparability with file fragment analysis, as the whole file needs to be analysed.

The third method, named File Header/Trailer (FHT) algorithm, also analysed the header and the footer, of the focus files. This time, the overall accuracy rate reached 95.83%. Later, Roussev and Garfinkel (2009) mentioned that this method is not suited for real-life FTI, as signatures do not always exist in the header/trailer, especially when dealing with file fragments in data recovery contexts.

The BFD approach was addressed by authors in (McDaniel and Heydari, 2003; Li et al., 2005), where they used the 1-g-distribution, which is a vector of two 256-element vectors, collecting blocks at the size of 20, 200, 500 and 1000 bytes. Contrary to common expectations, increasing the blocks lowered the result. While blocks of 20 bytes achieved a 100% classification result, 1000 bytes performed with only 77% worse. However, these blocks were not random parts of the file, but systematic fragments in the current block size. This approach did not extend the state-of-the-art on the research on fragmented FTI.

Karresand and Shahmehri (Dunham et al., 2005; Karresand and Shahmehri, 2006a,b) proposed a new centroid of 2-g based approach. They used the "Rate of Change" (ROC) that measures the difference of bytes. As such, file types with many 0 × 00 and 0xFF, such as JPEG's metadata, achieved the best result. The confusion matrices stated a true positive accuracy of 99.94%, unless no restart marker was within the fragment. In that case, the accuracy dropped to 42.66%. Their false positive rate was especially for high entropy files like ZIP and PE files by to 70%.

Table 1
Research Progress Table based on previous work.

Contributors	File/ Fragment	Method	#Types	#Files	Accuracy %
McDaniel and Heydari (2003)	File	BFA BFC FHT analysis	30	120	27.5 45.83 95.83
Li et al. (2005)	File	Manhattan distance Manhalanobis distance Multi-centroid	8 (5 classes)	800	82 (One-Centroid) 89.5 (Multi-Centroid) 93.8 (Example files)
Dunham at al. (2005)	File	Neural networks to classify encrypted data with the same key.	10	760	91.3
Karresand and Shahmehri (2006a,b)	Fragment	Oscar method (based on Mean and standard derivation of BFD) Biased for JPG	49	53	97.9 (JPG)
Karresand and Shahmehri (2006a,b)	Fragment	Oscar method + rate of change between consecutive byte values	51	57	87.3–92.1 (JPG 46–84 (ZIP) 12.6 (EXE)
Zhang et al. (Zhang and White, 2007)	Fragment	BFS and Manhattan distance	2	100	92.5
Moody and Erbacher (2008)	Fragment	Mean, standard deviation, kurtosis	8	200	74.2
Calhoun and Coles (2008)	Fragment	Fisher's linear discriminant, Statistical measurements	2	100	68.3–88.3 (bytes 129–1024) 60.3–86 (bytes 513–1024)
Amirani et al. (2008)	File	PCA + Neural networks feature extraction MLP Classifier	6	720	98.33
Cao et al. (2010)	File	Gram Frequency Distribution, Vector space model	4	1000	90.34 (2-g + 256 g as type signature)
Ahmed et al. (2010)	File	Cosine similarity, divide conquer, MLP classifier	10	2000	90.19
Ahmed et al. (Dhanalakshmi and Chellappan, 2009)	Both	Feature Selection, Content Sampling, KNN Classifier	10	5000	90.5 (40% of features) 88.45 (20% of features)
Amirani et al. (2013)	Both	PCA + Neural Networks feature extraction SVM Classifier	6	1200	99.16 (Whole files) 85.5 (1500 bytes fragments) 82 (1000 bytes fragments)
Evensen et al. (Daniel Evensen, 2015)	Both	*n*-gram analysis with naïve bias classifier	6	60000	99.51 (Whole files) 99.08 (8192 bytes fragments, 5 types) 98.34 (1024 bytes fragments, 5 types)
Beebe et al. (2016)	File	K-Means, Hierarchical classification	50	2600	57.56 (theorical model) 74.08 (winning model)
Bhatt et al. (2020)	Fragment	SVM, Hierarchical classification	14	14000	67.78

Dhanalakshmi and Chellappan (2009) researched a list of statistical measures for FTI. The approaches' accuracy of 66% was achieved in 14 out of 25 file types.

Ahmed and Lhee (2011) also made use of n-grams by investigating FTI on executable code within network packet streams. They determined that the order of n-grams influenced the accuracy. As a result, 3-g identify an executable with an accuracy of 92.78% (FP-rate: 4.69%, false negative rate of 2.53%). Subsequently, they used SVMs with an rbf-kernel for fast file type identification. Further research on n-grams was performed by Cao et al. (2010), who analysed the accuracy depending on the number of grams. They achieved a peak result using 300 g per fragment.

Other approaches were tested by Hall and Davis (Hall, 2006), who used the information entropy and compressibility to calculate the standard deviation of "sliding windows". The difference between known file type and the calculated plots, known as "goodness of fit", classified the file's type.

Another approach was used by Moody and Erbacher (Erbacher and Mulholland, 2007) in 2008. They developed a metrics-based approach named "SADI". The idea behind SÁDI is to focus on the statistics of encodings. They noted that all text-based file types, such as.txt,.html or.csv form a subset of file types, which could eventually be classified in a secondary analysis method. For the first stage, only a statistical analysis was used. The second stage then utilised pattern recognition algorithms.

Veenman (2007) subsequently experimented in 2008 with a new approach. He categorised files into the three major clusters: BFD, information entropy and Kolmogorov complexity. His findings indicated that higher entropies resulted in a loss of accuracy. For example, HTML and JPEG files were classified with 98% accuracy while ZIP files resulted in only 18% accuracy. This approach was extended by Calhoun and Coles (2008) without the use of metadata but by adding further statistical methods, such as Fisher's linear discriminant and the longest common subsequence.

Axelsson (2010) introduced the "Normalized Compression Distance" (NCD). The approach compared the compression of well-known file types to test fragments. A k-nearest-neighbour classifier's majority vote was used. Depending on k, different results were achieved: $k = 1$ resulted in an accuracy of 36.43%. $k = 10$ reached 32.86%.

Beebe et al. (2016) also mentioned a hybrid approach, which is a combination of the signature and statistical ones that can be applied during the multi-level class-to-type hierarchical classification process to optimise the file type identification in the context of file fragments.

2.3. Computational intelligence approaches

The third type of approach is Artificial Intelligence (AI)-based methods. Most common representatives of such systems are based on SVMs, NNs, Bayesian Networks (BNs) and kNN. Ahmed et al. (2010) studied several classifiers by using 1-g features. Their investigation on NNs, kNN and SVMs with kNN performed the best. In their research, they described a reduction in computational time

by a factor of fifteen in case a subset of features was used, and the sample blocks were randomly ordered. Li et al. (2010) worked also with SVMs with the linear kernel using all 28 1-g. They achieved an overall accuracy greater than 89%. Gopal et al. (2011) also utilized SVMs as well as kNNs on 1-g and 2-g comparing their accuracy to the commercial of the shelf (COTS) solutions: libmagic and TrID. To ensure, the signature was not classified, the first 512 bytes of each file were removed. Consequently, both the kNN on 1-g, and the SVM on 2-g classified with an accuracy of 85%, which is seven times better than the examined COTS. This approach as proven throughout several other scientific works (Young et al., 2012).

Sportiello and Zanero (2011) also focused on the feature selection of BFD, RoC, entropy, complexity, mean byte value for their SVM. According to their confusion matrices, they achieved 71.1% for.doc files using Entropy, BFD and Kolmogorov complexity. For Bitmap files, they utilized RoC to gain an accuracy value of 98.1%. Their approach was later repeated (Sportiello et al., 2012), checking for a different behaviour as all compound files, high entropy files like zips, were replaced by empty files. The different training phase resulted in three to five percent better test results. Another approach by Fitzgerald et al. (2012) made use of “natural language processing's (NLP) average contiguity between bytes of sequences and the longest continuous streak of repeating bytes” (Poisel et al., 2014) to archive a total accuracy of 40%, whereby the types according to the same group of types, as Moody and Erbacher (Erbacher and Mulholland, 2007) described, could vary. Because of that, the given approach might serve best to classify only certain groups of file types such as.csv,.gif, or.html. Several features, such as Hamming weight, standardized kurtosis, standardized skewness, or, maximum byte streak, to name a few, were used by Beebe et al. (2013) in combination with SVM classifiers. They achieved an accuracy of around 80% for the file type groups of text-based and multimedia file formats.

Amirani et al. (2013), (Amirani et al., 2008) used the Principal Component Analysis (PCA) to extract relevant features from the BFD of training files, an unsupervised learning process. They trained using a three-layered neural network (MLP) (Amirani et al., 2008) and an SVM (Amirani et al., 2013) in combination with a back-propagation algorithm. The resulting accuracy reached 99.16%, although other scientists (Poisel et al., 2014) complained about missing details on the training set. In addition, Carter (2013) who combined a disassembler and n-gram analysis to identify executable code. The classification was performed by a kNN. Kattan et al. (2010) came up with the idea to utilise Genetic Programming (GP) to FTI. They used the principles of segmentation, creation of file-prints and classification to FTI and achieved an accuracy of 70.77% for five file types.

Rainer Poisel et al. (2014) are not covered in detail as they are not relevant to this work. In short, the context considering approaches utilized meta-data about the information to gather an idea about its file types. The last group, “other approaches” is a collective of all approaches that could not be assigned to the other categories, included hybrid approaches of several other before mentioned approaches. So are the interpretation of binary visualization with SVMs (Contiet al., 2010) or Roussev and Garfinkel's (2009) combination of statistical and signature-based approaches typical examples for this category.

Beebe et al. (2016) presented a hierarchical classification model based on the clustering techniques for 50 file types. Their accuracy is up to 74.08%.

Bhatt et al. (2020) also used a hierarchy-based classification approach based on SVM to classify the file fragment. They gained the average accuracy of 67.78% and an F1-measure of 65% of 14 file types with 1000 fragments each.

3. Comprehensive study of n-gram analysis based on support vector machine and neural networks

Although the research topic of FTI has been studied, several approaches proposed new methods without testing them against all the previously existing methods. Consequently, some problems arise that require a detailed reappraise of previous approaches. In addition, the training and testing of NNs for FTI on public data corpora seem to be a gap in previous scientific work. Furthermore, the usage of SVMs with different kernels was not exhaustively performed. Hence, the leading questions are: how strong is the influence in results if a different setup of NNs and SVMs is used? How does it scale? To answer these questions, we describe an empirical study in this section and the experimental results in Section 4.

3.1. Dataset

Since almost all scientific work in this field is based on individual data sets, a direct comparison of the achieved results is impossible as described in (Garfinkel, 2010). To address this problem, some researchers have suggested the use of a public record. Hence, in our work, all experiments are carried out with a public data corpus called the Realistic Data Corpus, aka “RealisticDC” or “RealDC” (http://downloads.digitalcorpora.org/corpora/files/govdocs1/) was developed by the Digital Intelligence and Investigation Directorate (DiiD) of CERT in the Software Engineering Institute at the Carnegie Mellon University. We look at the file type package of RealDC datasets. In this package, all files have been sorted by type. These include the most popular types:.xls,.doc,.csv,.txt,.jpg and.ppt. These files are chosen because of their popularity and also the experimental results from these file types can be extended for other relevant file types, e.g. .png,.html, etc. These files were then pre-processed (carved, labelled) for the following experiments.

3.2. Experiment 1: n-gram classification using a NN

The first experiment used a NN for the classification of file types. *n-grams* were fed into a multi-layer NN for learning. During the experiment, the learning rate for optimum results was determined by using an exhaustive search.

3.2.1. Data preparation and training

In this experiment, RealDC was split into two groups, one for testing and one for training. The most popular file types included in the RealDC are used, containing the formats.xls,.doc,.csv,.txt,.jpg and.ppt. For the selection of the files, only files with a file size greater than or equal to 5200 bytes were used. The first 200 bytes were cut off, preventing any magic-byte/file header from being recognized. The following 5000 bytes were used for analysis. Other bytes are ignored. Note that the cut-off size also depends on the file types, the size we used in this paper is just for the relevant file types in our experiments. We then convert the byte frequency distribution of different file types to n-grams with $n = 1$ and 2. The data is also normalized and then fed into the NN's input layer (256 nodes). Our NN has 256 neurons on input layer, 128 neurons in the hidden layer and six neurons for the last, the output layer (equivalent to 6 file types).

We applied cross validation and trained with 6000 files (1000 files for each file types) whereas 6000 files for the testing (1000 files for each file types). We also optimised the learning rate by varying it from 0 to 99% and selecting the best successful rate. The optimal rate for $n = 1$ is 1% and for $n = 2$ is 8.17%.

3.3. Experiment 2: n-gram classification using an SVM

In the second experiment, the same data corpus was used for training a SVM. Therefore, only the comparable n-grams from the previous experiment, n = 1 and n = 2, were tested. Both, linear kernel and radial basis function (rbf) kernel were trained and tested on disjunctive test/train data corpora.

3.3.1. Experimental setup

For setting up the experiment, "WEKA" (https://www.cs.waikato.ac.nz/ml/weka/) version 3.8.3 was used. The training data from Experiment 1 were also used to train four different SVMs. Two SVMs were trained using a linear kernel (Vapnik et al., 1997), the other two were trained with rbf kernel (Cristianiniand and Shawe-Taylor, 2000). Each kernel was trained and tested once for n = 1 and for n = 2.

3.3.2. Data preparation and training

In detail, each file type was classified once against other file types. Thus,.csv-files were trained against.doc,.jpg,.ppt,.txt and.xls. Then,.doc-files were trained against.jpg,.ppt,.txt and.xls and so on. In total, the SVM trained 15 times.

4. Comparison of experimental results and discussion

After six experiments, two NNs and four different SVMs were trained on the same set of data. For each experiment, a "Confusion Matrix" was calculated, listing the prediction per file type in rows of data. The correct file type is labelled in the last column, next to the total classification result. The correct classifications were marked in green. The testing results in this section are True Positive rate.

4.1. Experimental results: neural network

4.1.1. Testing results for neural network n = 1

The confusion matrix (Table 2) visualizes a significant drop of classification success for.ppt files. Surprisingly, the file types.csv,.txt and.xls were classified correctly in most cases, although these formats all contain much ASCII-encoded symbol.

4.1.2. Testing result for neural network n = 2

On the second neural network, Table 3 shows results were computed by feeding the input-layer with all attributes (2562). As an overall classification, a result of 85.88% (in average for 6 testing file types) was achieved. For the investigational purpose of this result, a progress graph was drawn, showing the average success rate within the next 100 classifications on the y-axis and the current sample data on the x-axis (Fig. 1).

The red area at the beginning of this graph indicates the testing results from the training. This area was not counted into the overall result as test and train data corpus needs to be disjoint. It is particularly noticeable that the classification of jpg files did not stay reliably high but reduced suddenly. This point should be examined further in the future work and so does the.ppt file type.

4.2. Experimental results: support vector machine

This section presents the outcome of four experiments on support vector machines. There are confusion matrices for each experiment, thereby making the result comparable to the neural network's result. If the same n was used in the description, identical attributes on both tests can be presupposed.

4.2.1. Testing result using a linear kernel with n = 1

Testing all 5241 files from test data corpus, 4795 files were classified correctly. So, the overall classification ratio was 91.49%. The remaining 446 files, 8.50%, were misidentified. The following confusion matrix (Table 4) lists the classification results in detail.

The biggest flaw occurred at ppt-files. They failed in 24.7% of all classifications. Apart from those files, this test proved solid results, in most cases at ~95%.

4.2.2. Testing result using rbf kernel with n = 1

While initiating the classification on the test data corpus, the flaws in training became clearly visible. The overall classification stagnated at 65.12%, failing in 34.87% of all cases.

As shown in the confusion matrix (Table 5), the file types ppt and txt were not classified correctly in general. This observation matches the observations during the training and might indicate a deficiency in the classifier or the training set. Without these two file types, the classification ratio had been on 91.82%, potentially outperforming the linear kernel.

4.2.3. Testing result using a linear kernel with n = 2

After finishing the experiments for n = 1, linear and radial basis function kernels were tested for n = 2. During the training phase for the linear kernel, a classification result of 100% was achieved. After this perfect classification training, the linear kernel SVM classified 4756 files out of 5241 files correctly. The total success rate was 90.74%. In detail, especially.csv,.doc, and.xls performed above 95%. Like all previous experiments with SVMs, ppt performed worst, this time with only 77% accuracy (Table 6).

4.2.4. Testing result using a rbf kernel with n = 2

In the last experiment, the trained SVM was tested using the test data corpus. The total number of attributes was 2562, again. In the

Table 2
Confusion matrix Neural Network n = 1.

csv	doc	jpg	ppt	txt	xls		
716	1	0	1	1	22	96.62 %	csv
0	761	33	92	3	2	85.41 %	doc
0	1	868	15	9	4	96.76 %	jpg
0	383	24	476	8	1	53.36 %	ppt
0	2	10	80	802	0	89.70 %	txt
47	1	0	0	0	850	94.65 %	xls

Table 3
Confusion matrix Neural Network n = 2.

csv	doc	jpg	ppt	txt	xls		
735	0	0	1	1	4	99.19 %	csv
0	609	2	242	33	2	68.58 %	doc
0	0	786	28	72	7	88.01 %	jpg
1	184	16	681	11	0	76.25 %	ppt
0	1	2	63	831	0	92.64 %	txt
65	1	0	0	0	833	92.65 %	xls

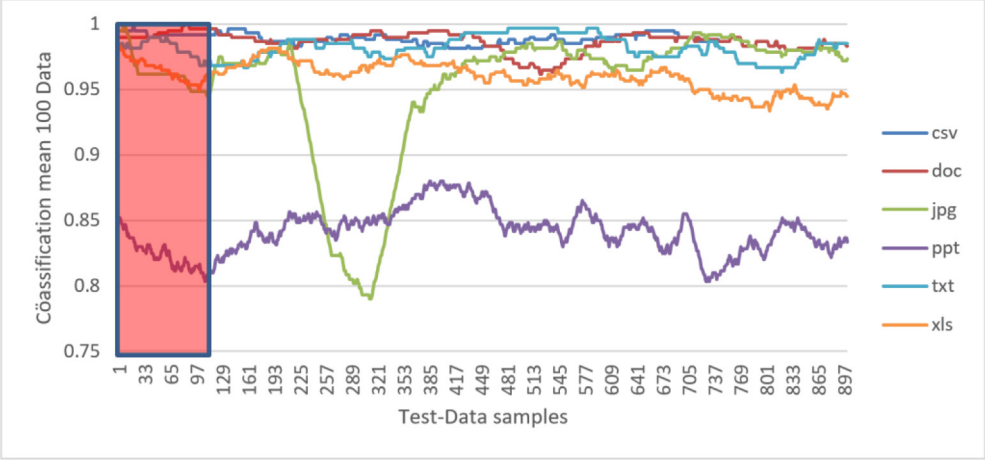

Fig. 1. Classification result of testing datasets for different file types.

Table 4
Confusion matrix SVM with linear kernel on n = 1.

csv	doc	jpg	ppt	txt	xls		
708	0	1	1	30	1	csv	95.5 %
2	878	0	10	5	5	doc	97.6 %
0	8	806	86	0	0	jpg	89.6 %
1	10	199	681	0	9	ppt	75.7 %
37	7	2	1	852	1	txt	94.7 %
0	6	4	20	0	870	xls	96.7 %

end, 4450 files were classified correctly. That makes 84.90% overall classification, or 15.09% falsely classified files. Again, ppt-files were poorly classified. A property that was observed in all tests in this series (Table 7).

4.3. Evaluation of results

Summarizing the experiments, the results can be visualized and interpreted in several ways. First, a general overview of the testing results realized is displayed in Fig. 2.

Table 5
Confusion matrix SVM with rbf kernel on n = 1.

csv	doc	jpg	ppt	txt	xls		
657	17	0	0	66	1	csv	88.66 %
0	849	1	0	5	45	doc	94.33 %
0	7	879	7	0	7	jpg	97.66 %
1	34	714	69	2	80	ppt	07.66 %
338	383	0	0	179	0	txt	19.88 %
0	110	2	4	4	780	xls	86.66 %

Table 6
Confusion matrix SVM with linear kernel on n = 2.

csv	doc	jpg	ppt	txt	xls		
602	1	0	0	137	1	csv	81.24%
0	802	0	3	94	1	doc	89.11%
0	21	727	100	52	0	jpg	80.77%
0	18	161	660	58	3	ppt	73.33%
15	47	0	0	838	0	txt	93.11%
0	16	0	3	60	821	xls	91.22%

Table 7
Confusion matrix SVM with rbf kernel on n = 2.

csv	doc	jpg	ppt	txt	xls		
710	2	0	0	28	1	csv	95.81 %
0	884	0	5	11	0	doc	98.22 %
7	36	750	105	2	0	jpg	83.33 %
0	8	185	696	3	8	ppt	77.33 %
35	30	0	1	834	0	txt	92.66 %
0	15	0	3	0	882	xls	98.00 %

The overall figure is split in half (Table 8), showing the difference for n = 1 on the left and n = 2 on the right. Except for the gap in RBF-kernel's, the neural network and the support vector machines predicted the results in an insignificantly different way. The SVMs accuracy ranged from 65.12% to 91.49% while the NN's accuracy ranged from 85.80% to 85.88%. The results indicate that at least one Support Vector Machine outmatched the neural network approach. Comparing the overall results (Fig. 2), with the intention of answering one of the initial problems, to estimate the evolution of accuracy for increasing n, two answered can be formulated. First, the overall accuracy did increase from 80.7% to 87.1% by increasing n by one. Second, the increase of accuracy by only 0.1% on the neural network was not significant. And the linear kernel on the SVM even decreased the accuracy. Contrary to expectations, the increase of computational power by increasing the size of the neural network, the n = 2 approach did not perform better than the monograms did. It seems to be a paradox, which also concerned Li et al. (2005), who already noted an inconsistency with n-gram analysis, as increasing data input reduced the accuracy.

Interpreting the confusion matrices, some tests revealed typical fail predictions within the predicted file types. As an example, SVMs mis-predicted ppt file as often as jpg files and vice versa that is an interesting behaviour, which might indicate a related file structure. So did the neural network approach prefer to falsely predict ppt files as doc files and vice versa. Different types of errors can draw the conclusion, that both algorithms found a unique way to solve the problem, although it is probably impossible to fully reconstruct the exact behaviour inside the neural network. To investigate the different approaches by another perspective, the evolution of testing to training accuracy will be highlighted.

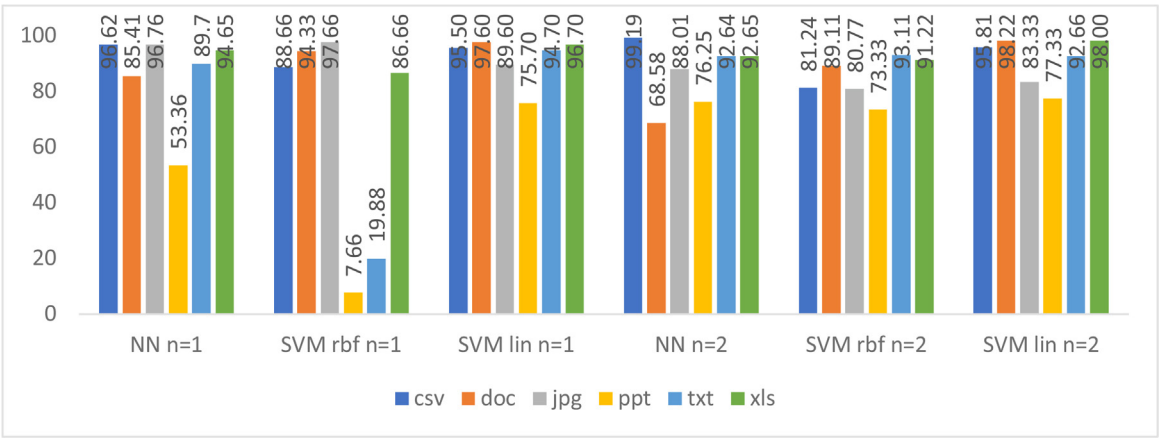

Fig. 2. Overall testing results.

Obviously, the training achieved always a better accuracy than the testing.

Fig. 3 through 6 visualize the evolution by the end of the training to the end of the testing stage. Of particular significance is the linear kernel's training set. A 100% perfect classification might indicate possible overfitting of the SVM.

In this case, a worse testing result on a disjunctive data corpus should validate this result. But the testing of this approach did not confirm this theory. It rather seems to be better of having the equality in all file type predications except on JPG. A training set of only 1000 files per type seems to be small enough to allow this theory, especially when the number of attributes is 2562.

4.4. Discussion of results and comparison to alternative approaches

As mentioned above, the linear kernel SVM performed better than the neural network. Yet, the results for the SVMs are widely spread and general superiority of SVMs over NN cannot be claimed. In both cases, high classification accuracies were noticed. Also, the expected growth on accuracy did not correlate with the increase of n.

Mayer (2011) worked on the same performance problems as this work. He achieved only 66% but as 25 file types were used, the work is not exactly comparable. Yet, Ahmed and Lhee (2011) use the n-gram approach with n = 3. That leads to the assumption, that increasing n also increases the meaningfulness of the gram. However, there is some dissent on this topic. Evensen et al. (Amirani et al., 2013) noted the following: "One important observation

Table 8
Testing results split into features versus classifier.

	$n = 1$	$n = 2$
NN	85.8%	85.9%
SVM rbf	65.1%	84.9%
SVM lin	91.4%	90.7%
Overall	80.7%	87.1%

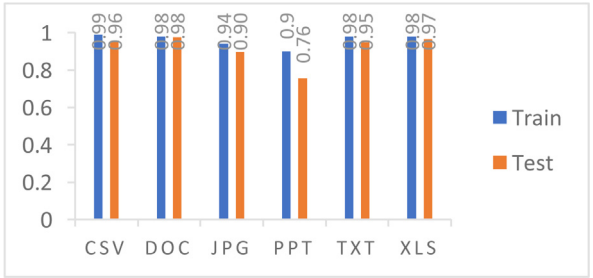

Fig. 3. SVM n = 1 linear kernal.

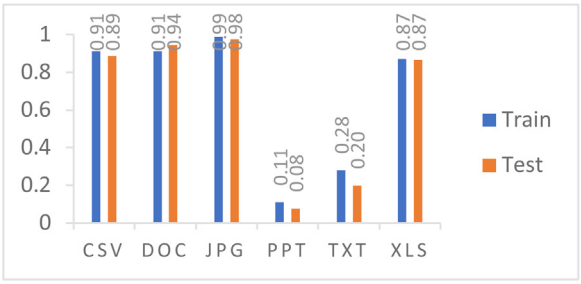

Fig. 4. SVM n = 1 rbf kernal.

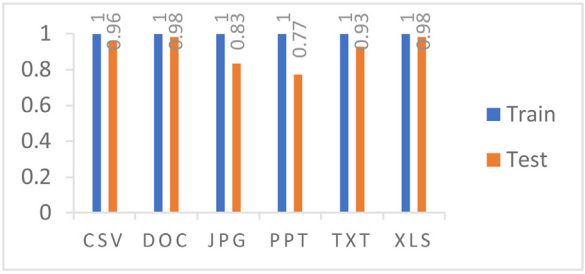

Fig. 5. SVM n = 2 linear kernal.

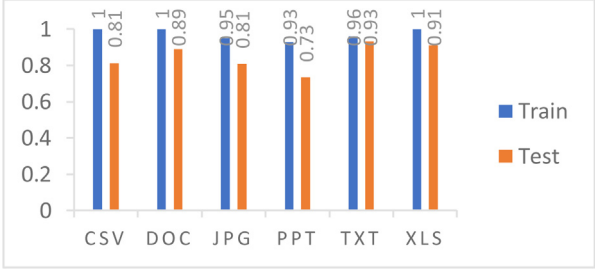

Fig. 6. SVM n = 2 rbf kernal.

done here is the fact that when considering around 350 files for learning, 3-g is not superior to 2-g"

Cao et al. (2010) delivered some proof to support Mayer. Their result was an optimum for n = 300. This seems to be a theoretical value, as it is not practically possible to feed a neural network with 2300 input neurons, even if there would be no hidden, but only one output neuron at all. Probably one of the closest comparable approaches was the work of Gopal et al. (2011), who also used SVMs with 1- and 2-g. They achieved results on the SVMs TP-rate >85%.

Comparing the results with Beebe et al. results in (Beebe et al., 2013), also the linear kernel SVM approach improved on the RBF kernel. However, the used feature was not implemented the same way, and, in their results, the bigrams outbid the Unigrams. In general, the linear kernel seems to work better in combination with n-gram approaches, even if they are slightly modified.

Comparing the results with the results by Mehdi et al. in (Amirani et al., 2013), the classifiers can be compared instead of the features. In their scenario, neural networks and support vector machines had a genuinely neck-and-neck race. Both achieved classification rates beyond 98%. Although this is a clear indicator for a better-suited feature, both, neural networks and support vector machines did not clearly differentiate. In their work, neural networks were even slightly better than SVMs.

The most detailed confusion matrices were reported by Wang et al. (2018). The amount of information produced by their research allows us to investigate the reported problem of fail classifications of.ppt files as.jpg. Focusing on the.ppt and.jpg problem, they also had a significant fail-prediction of.jpg files as.ppt and.pptx. Since this problem has now occurred in two research studies with different characteristics, it seems reasonable to conclude that these two file types are very similar or include each other.

Also, Li et al. (2010) used all monograms as input for the SVM. The resulting accuracy, 89%, fits perfectly to the best experiment's result of 91.4%.

Comparing the results achieved with equivalent scientific works, neither the change of data corpus nor the application of neural networks changed the classification accuracy significantly. That leads to the conclusion, that the feature of *n-gram* analysis does not allow any better classification than the mean of all scientific works on that approach. The results were in between 85% and 92%. To further increase the accuracy, an increase for n must be performed without raising the issue of scalability.

5. Conclusion

In this paper, we perform empirical research for the systematisation of knowledge on the performance of two popular machine learning-based approaches for FTI: SVMs and NNs for n-grams analysis. Although the limit for n could be increased due to additional computation power in the near future, the most important limitation is the fact that the method scales exponentially. Consequently, the development of n-gram-based approaches, that perform better with scalability or shift the algorithm's computational cost towards the need for memory usage, is imperative. Although it was not possible to give a clear preference to SVMs, bad scalability for NNs was proven.

Looking to the future, the next decade is likely to see the results of ongoing research and development into computational intelligence deep learning techniques (Nguyen Thi et al., 2017; Kuppa et al., 2018) can be applied in the file identification. Also, the implementation of cloud-based AI and continuously hashing of known file fragments are likely going to increase the accuracy of FTI, as well as decrease the computational time and memory usage. Also, ongoing research could help to improve the functionality of neural networks if we can explain them (Kuppa and Le-Khac, 2020).

References

Ahmed, I., Lhee, K., Nov. 2011. Classification of packet contents for malware detection. J. Comput. Virol. 7 (4), 279–295.

Ahmed, I., Lhee, K., Shin, H., Hong, M., 2010. Content-based file-type identification using cosine similarity and a divide-and-conquer approach. IETE Tech. Rev. 27 (6), 465.

Amirani, M.C., Toorani, M., Beheshti, A., 2008. A new approach to content-based file type detection. In: 2008 IEEE Symposium on Computers and Communications, Marrakech, pp. 1103–1108.

Amirani, M.C., Toorani, M., Mihandoost, S., 2013. Feature-based type identification of file fragments: feature-based type identification of file fragments. Secur. Commun. Network. 6 (1), 115–128. Jan. 2013.

Axelsson, S., 2010. The Normalised Compression Distance as a file fragment classifier. Digit. Invest. 7, S24–S31. Aug. 2010.

Beebe, N.L., Maddox, L.A., Liu, L., Sun, M., 2013. Sceadan: using concatenated N-gram vectors for improved file and data type classification. IEEE Trans. Inf. Forensics Secur. 8 (9), 1519–1530. Sep. 2013.

Beebe, N., Liu, L., Sun, M., 2016. Data type classification: hierarchical class-to-type modeling. In: 12th IFIP International Conference on Digital Forensics (DF), Jan 2016, New Delhi, India, pp. 325–343. https://doi.org/10.1007/978-3-319-46279-0.

Bhatt, M., Mishra, A., Kabir, M.W.U., Blake-Gatto, S.E., Rajendra, R., Hoque, M.T., Ahmed, I., 2020. Hierarchy-based file fragment classification. Mach. Learn. Knowl. Extr. 2, 216–232.

Breitinger, F., Stivaktakis, G., Baier, H., 2013. FRASH: a framework to test algorithms of similarity hashing. Digit. Invest. 10, S50–S58, 2013.

Calhoun, W.C., Coles, D., 2008. Predicting the types of file fragments. Digit. Invest. 5, S14–S20. Sep. 2008.

Cao, Ding, Luo, Junyong, Yin, Meijuan, Yang, Huijie, 2010. Feature selection-based file type identification algorithm. In: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, Xiamen, China, pp. 58–62.

Carter, J.M., 2013. Locating executable fragments with Concordia, a scalable, semantics-based architecture. In: Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop on - CSIIRW '13, Oak Ridge, Tennessee.

Chawathe, S.S., 2009. Effective whitelisting for filesystem forensics. In: 2009 IEEE International Conference on Intelligence and Security Informatics, Richardson, TX, USA, 2009, pp. 131–136.

Collange, S., Dandass, Y.S., Daumas, M., Defour, D., 2009. Using graphics processors for parallelizing hash-based data carving. In: 42nd Hawaii International Conference on System Sciences, Hawaii, USA, 2009, pp. 1–10.

Conti, G., et al., 2010. Automated mapping of large binary objects using primitive fragment type classification. Digit. Invest. 7, S3–S12. Aug. 2010.

Cristianiniand, Nello, Shawe-Taylor, John, 2000. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, 2000.

Dandass, Y.S., Necaise, N.J., Thomas, S.R., 2008. An empirical analysis of disk sector hashes for data carving. J. Digit. Forensic Pract. 2 (No 2).

Daniel Evensen, John, 2015. Clustered File Type Identification.Pdf. Master Thesis. University of Agder.

Dhanalakshmi, R., Chellappan, C., 2009. File format identification and information extraction. In: 2009 World Congress on Nature & Biologically Inspired Computing. NaBIC, Coimbatore, India, pp. 1497–1501, 2009.

Du, X., Ledwith, P., Scanlon, M., August 2018. Deduplicated disk image evidence acquisition and forensically-sound reconstruction. In: 17th IEEE International Conference on Trust, Security And Privacy In Computing And Communications. TrustCom-18, New York, USA.

Dunham, J.G., Sun, Ming-Tan, Tseng, J.C.R., 2005. Classifying file type of stream ciphers in depth using neural networks. In: 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005., Cairo, Egypt, 2005, pp. 512–518.

Erbacher, R.F., Mulholland, J., 2007. Identification and localization of data types within large-scale file systems. In: Second International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE'07), Bell Harbor, WA, pp. 55–70.

Fitzgerald, S., Mathews, G., Morris, C., Zhulyn, O., 2012. Using NLP techniques for file fragment classification. Digit. Invest. 9, S44–S49. Aug. 2012.

Garfinkel, S.L., 2006. Forensic feature extraction and cross-drive analysis. Digit. Invest. 3, 71–81. Sep. 2006.

Garfinkel, S.L., 2010. Digital forensics research: the next 10 years. Digit. Invest. 7, S64–S73. Aug. 2010.

Garfinkel, S.L., McCarrin, M., 2015. Hash-based carving: searching media for complete files and file fragments with sector hashing and hashdb. Digit. Invest. 14, S95–S105. Aug. 2015.

Garfinkel, S., Nelson, A., White, D., Roussev, V., 2010. Using purpose-built functions and block hashes to enable small block and sub-file forensics. Digit. Invest. 7, S13–S23. Aug. 2010.

Gionis, A., Indyk, P., Motwani, R., 1999. Similarity search in high dimensions via hashing. In: Proceeding VLDB '99 Proceedings of the 25th International Conference on Very Large Data Bases, Sep 1999, pp. 518–529.

Gopal, S., Yang, Y., Salomatin, K., Carbonell, J., 2011. Statistical learning for file-type identification. In: 10th International Conference on Machine Learning and Applications and Workshops, Honolulu, HI, USA, 2011, pp. 68–73.

Hall, G.A., 2006. Sliding Window Measurement for File Type Identification. IEEE workshop on Information Assurance Workshop, 2006.
Karresand, M., Shahmehri, N., 2006a. Oscar — file type identification of binary data in disk clusters and RAM pages. In: Fischer-Hübner, S., Rannenberg, K., Yngström, L., Lindskog, S. (Eds.), Security and Privacy in Dynamic Environments, vol. 201. Springer US, Boston, MA, pp. 413–424, 2006.
Karresand, M., Shahmehri, N., 2006b. File type identification of data fragments by their binary structure. In: 2006 IEEE Information Assurance Workshop, West Point, NY, 2006, pp. 140–147.
Kattan, A., Galván-López, E., Poli, R., O'Neill, M., 2010. GP-fileprints: file types detection using genetic programming. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (Eds.), Genetic Programming, vol. 6021. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 134–145, 2010.
Kornblum, J., 2006. Identifying almost identical files using context triggered piecewise hashing. Digit. Invest. 3, 91–97. Sep. 2006.
Kuppa, A., Le-Khac, N.-A., 2020. Black box Attacks on explainable artificial intelligence (XAI) methods in cyber security. In: IEEE International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, July, 2020. https://doi.org/10.1109/IJCNN48605.2020.9206780.
Kuppa, A., Grzonkowski, S., Le-Khac, N.-A., 2018. Enabling trust in deep learning models: a digital forensics case study. In: 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications. TrustCom-18, NY, USA. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00172. Aug. 2018.
Li, Wei-Jen, Wang, Ke, Stolfo, S.J., Herzog, B., 2005. Fileprints: identifying file types by n-gram analysis. In: Proceedings from the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2005., West Point, NY, USA, pp. 64–71.
Li, Q., Ong, A.Y., Suganthan, P.N., Thing, V.L.L., 2010. A novel support vector machine approach to high entropy data fragment classification. In: Proceedings of the South African Information Security Multi-Conference (SAISMC 2010), p. 10.
Lillis, D., Breitinger, F., Scanlon, M., 2018. Expediting MRSH-v2 approximate matching with hierarchical Bloom filter trees. In: Digital Forensics and Cyber Crime. ICDF2C 2017. Springer, Cham. https://doi.org/10.1007/978-3-319-73697-6_11.
Mayer, R.C., 2011. FILETYPE IDENTIFICATION USING LONG, SUMMARIZED N-GRAMS. Master Thesis. Naval Postgraduate School, Monterey, California. March 2011.
McDaniel, M., Heydari, M.H., 2003. Content based file type detection algorithms. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9, vol. 9. January 2003.
Moody, S.J., Erbacher, R.F., 2008. Statistical analysis for data type identification. In: 2008 Third International Workshop on Systematic Approaches to Digital Forensic Engineering, Oakland, CA, USA, 2008, pp. 41–54.
Nguyen Thi, N., Cao, V., Le-Khac, N.-A., 2017. One-Class Collective Anomaly Detection Based on LSTM-RNNs, vol. 10720. Springer Transactions on Large-Scale Data and Knowledge-Centered Systems, LNCS, pp. 73–85. https://doi.org/10.1007/978-3-662-56266-6.
Pal, A., Memon, N., 2009. The evolution of file carving. IEEE Signal Process. Mag. 26 (2), 59–71. Mar. 2009.
Poisel, R., Rybnicek, M., Tjoa, S., 2014. Taxonomy of data fragment classification techniques. In: Gladyshev, P., Marrington, A., Baggili, I. (Eds.), Digital Forensics and Cyber Crime, vol. 132. Springer International Publishing, Cham, pp. 67–85, 2014.
Roussev, V., 2011. An evaluation of forensic similarity hashes. Digit. Invest. 8, S34–S41. Aug. 2011.
Roussev, V., Garfinkel, S.L., 2009. File Fragment Classification-The Case for Specialized Approaches. 4th IEEE Workshop on Systematic Approaches to Digital Forensic Engineering, Berkeley, California, USA, pp. 3–14, 2009.
Roussev, V., Quates, C., 2012. Content triage with similarity digests: the M57 case study. Digit. Invest. 9, S60–S68. Aug. 2012.
Roussev, V., 2010. Data fingerprinting with similarity digests. In: Chow, K.-P., Shenoi, S. (Eds.), Advances in Digital Forensics VI, vol. 337. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 207–226, 2010.
Roussev, V., Quates, C., Martell, R., 2013. Real-time digital forensics and triage. Digit. Invest. 10 (2), 158–167. Sep. 2013.
Ruback, M., Hoelz, B., Ralha, C., 2012. A new approach for creating forensic hashsets. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics VIII, vol. 383. Springer Berlin Heidelberg, pp. 83–97, 2012.
Shannon, C.E., 1948. A mathematical theory of communication. Bell Syst. Techn. J. 1948 55.
Sportiello, L., Zanero, S., 2011. File block classification by support vector machine. In: 2011 Sixth International Conference on Availability, Reliability and Security, Vienna, Austria, 2011, pp. 307–312.
Sportiello, L., Zanero, S., 2012. Context-based file block classification. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics VIII, vol. 383. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 67–82, 2012.
Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York, 1998.
Vapnik, V., Golowich, S., Smola, A., 1997. Support vector method for function approximation, regression estimation, and signal processing. In: Mozer, M., Jordan, M., Petsche, T. (Eds.), Advances in Neural Information Processing Systems, vol. 9. MIT Press, Cambridge, MA, pp. 281–287, 1997.
Veenman, C.J., 2007. Statistical disk cluster classification for file carving. In: Proceedings of the First International Workshop on Computational Forensics, Manchester, UK, 2007.
Wang, F., Quach, T.-T., Wheeler, J., Aimone, J.B., James, C.D., 2018. Sparse coding for N-gram feature extraction and training for file fragment classification. IEEE Trans. Inf. Forensics Secur. 13 (10), 2553–2562. Oct. 2018.
Young, J., Foster, K., Garfinkel, S., Fairbanks, K., 2012. Distinct sector hashes for target file detection. Computer 45 (12), 28–35, 2012.
Zhang, L., White, G.B., 2007. An approach to detect executable content for anomaly based network intrusion detection. In: 2007 IEEE International Parallel and Distributed Processing Symposium, CA, USA, 2007, pp. 1–8.