File type identification tools for digital investigations

Adrien Dubettier, Tanguy Gernot, Emmanuel Giguet, Christophe Rosenberger

File Type Identification Tools for Digital Investigations

Adrien Dubettier[a], Tanguy Gernot[a], Emmanuel Giguet[a], Christophe Rosenberger[a]

[a]*Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France*

Abstract

Digital forensics is the process of identifying, preserving, analyzing, and documenting digital evidence for investigation purposes. Building or using file analysis tools is of great interest for a forensic expert to collect high-level information in a short time. In this paper, we consider the examination of files contained in digital media, especially files with possible incorrect types. This often reveals a simple way to hide sensitive content such as porn images, passwords, or accounts. Many commercial and free forensic tools are available for file type identification (FTI). In this work, we assess the performance of ten of them on two significant datasets and scenarios. The main issue we address is the relevance of the tools for forensic purposes. The underlying question is: do expectations meet reality? Our experiments highlight the significant disparity in the accuracy and behavior of the studied tools.

Keywords: Digital Forensics, Digital Evidence Assessment, Comparative Evaluation of Forensics Tools, Benchmarking, File Type Identification, File Systems.

1. Introduction

The scope of Digital Forensics covers the application of forensically sound technologies and methods that deal with the recovery, the investigation, the analysis and the reporting on digital materials and digital traces stored in electronic devices. While Digital Forensics has been raised in the late twentieth century with the increasing prevalence of digital technologies for conducting criminal activities [1], its scope is now extended to legal activities including Art and Cultural Heritage where digital devices may contain material and traces related to the authors, their work, their creation process, their correspondence [2, 3, 4]. Whatever the field of application of Digital Forensics, investigative technologies and tools must meet reliability and accuracy requirements [5]. Their performance should be assessed in terms of speed, but also in terms of accuracy, using well-defined formal assessment procedures, in a way to reinforce trust [6, 7]. This is precisely one of the objectives of Digital Forensic Science: contributing to the professionalization of Digital Forensics by providing objective and independent evaluation protocols in order to lead to reproducible results [8, 9, 10]. This aim leads to define evaluation protocols, with explicit guidelines, clear evaluation metrics, and dedicated datasets.

In an operational context, a forensic expert needs to use one or more tools to quickly process files stored in a device. Many commercial off-the-shelf (COTS) products are proposed and many free open-source software are also candidates for this task. It is often difficult to choose the appropriate tools considering performance (computation time), accuracy, cost and usability. These criteria have to be qualified through a rigorous protocol. Furthermore, it is

now popular *to blend the models*, that is to combine the output of different solutions to produce a more robust classifier.

In this article, we are interested in the comparative evaluation of file type identification (FTI) tools. After the forensic acquisition process, which includes device write-blocking, device identifier recording and bit-by-bit imaging, determining file types is one of the first investigation processes carried by investigators at an early stage. With the ever-growing need for data storage which leads to increasingly large storage devices, mining the content of disks requires the use of automated tools. In fact, the investigator has to consider devices ranging from one USB stick, to smartphones, laptops, personal computers, hard drives, and terabytes of server disk drives, in order to recover information or to search for evidence [11]. Automatically determining file types permits to get a quick overview of the storage device content, sorted or filtered by file categories (e.g. office documents, photos, videos, mailboxes), in order to help the investigator [12, 13]. As for example, file type identification is used to locate music in copyright infringement cases or to identify images and videos in child pornography cases. It may also help find office documents in fraud cases.

In Windows-like environment, which is still common, file types can be derived from the filename extension [14]. This technique can be considered as a convenient way to embed the file type in the filename, and then, to ease its recognition by the user and to facilitate its opening by an appropriate software. However, in such environments, changing file extensions is also a very convenient and easy way that permits to hide illegal activities with the least effort: once the extension is renamed, basic file type identification fails. An image file, for instance, is not recognized as an image anymore. To counter this concealment, several investigative tools have been proposed to determine the real media type of a file, regardless of its extension.

In this work, we selected 10 FTI solutions. The first one is the classical Unix command *File* as baseline solution. We selected four (4) open-source software freely available on the Internet and of various popularity, namely *filetype* (311 stars on GitHub), *file-type* (2.2k stars on GitHub), *detect-file-type* (11 stars on GitHub), *guess-file-type* (4 stars on GitHub) [15, 16, 17, 18]. We also considered four (4) professional solutions: *fidentify*, *TrID*, *EnCase* and *Autopsy* [19, 20, 21, 22]. We compare them in terms of accuracy and computing performance in order to find the most efficient ones. Our objective is to help investigators and researchers choosing the one that suits their needs. Moreover, we are also interested to better understand their limitations to be enhanced in the future.

The contributions of the paper are the following:

- A large-scale literature review of existing tools and methods for file type identification is provided. We consider both academic and non-academic forensic tools, including commercial tools used in digital investigation cases;
- A rigorous protocol, involving two significant complementary datasets and performance metrics for the comparative study of file identification tools, is introduced;
- One dataset has been specifically built in order to address the coverage issue of the tools regarding the number of file types properly taken into account. It contains 17500 files with 110 different file extensions;
- A large dataset composed of 1 million files with 63 different file extensions from Digital Corpora [23] is used in order to simulate a real case and assess execution time performance;
- Two concealment strategies have been worked out as scenarios: randomly altering all file extensions, systematically removing them;
- An in-depth analysis is conducted to identify the parameters that impact the performance of the different tools. A simple experiment shows that combining currently available tools permits to significantly improve the accuracy rate, from +0.2% to +5.9% on the two evaluation datasets;
- This work demonstrates the benefits of benchmarking forensic tools independently and rigorously.

The paper is organized as follows. In section 2, we introduce the different ways to identify the file type, including file extensions and binary signatures. In section 3, we provide a detailed description of the comparative protocol

we designed for this study, including tested tools, datasets and performance metrics. We show in section 4 the performance achieved by the tools when applied on the two datasets. We conclude and give some perspectives in section 5.

2. Background

A file can be defined as information represented by a byte sequence, organized according to a convention or format, proposed by an author or a consortium. Files are designated by an external filename that is usually composed of two parts in Windows-like environments: the name which allows a user to have access to it, and the extension that signifies the file type. With the extension, the operating system can determine which software to launch to open the file: a file association table maps file extensions to applications. Using file extension is a simple way to identify common file types. Indeed, some extensions are so common that they can reliably be used to categorize a file: The probability that these extensions match other file types is very low. For instance, we may think about exe, which refers to an executable file on Windows, txt which designates a plain text file, or jpg, png and gif which are common image formats. Today, nobody would use these extensions to refer to different file types. However, there are many limits when relying on file extensions to categorize files. (1) Several extensions can refer to the same file type, e.g. jpg and jpeg. (2) Some operating systems do not rely on file extension and do not require the presence of a file extension to launch the appropriate application: unlike Windows, MacOS lookup in a registry to find the association between a file and an app. (3) Naming convention conflicts may exist between apps that may use the same extension for different purposes. (4) As mentioned previously, an individual can easily and intentionally rename file extensions to prevent the automatic categorization of files.

To address the issues of variable naming conventions, and the possible absence of file extensions when exchanging files or information, the MIME standard (Multipurpose Internet Mail Extensions) has been introduced. It defines the structure of E-Mails and Internet messages and it includes a Content-Type field which provides a standardized way to declare media types. For instance, jpeg images are associated with the image/jpg Content-Type whatever the extension (empty, jpg, jpeg, or any other string). Media types are managed by the Internet Assigned Numbers Authority (IANA) and an official list of registered media types is provided (See https://www.iana.org/assignments/media-types/media-types.xhtml). File extension and MIME content types are exogenous metadata that gives information about the media type. They can be altered without corrupting the file itself. The file content is not affected by the change and the file is still readable by the appropriate application.

Another way to detect more reliably the file type is to look for the presence of a known file type signature in the file. Such signatures are called "magic numbers" (See Wikipedia's List of file signatures at https://en.wikipedia.org/wiki/List_of_file_signatures, File Signatures website at https://filesignatures.net/, or Gary Kessley's file signature tables at https://www.garykessler.net/). When they are present, magic numbers are embedded in the binary representation of the file, often at the beginning of the file like any header, and they appear as a fixed string of bytes. They sometimes have a meaning, such as transcribing the name of the format in ASCII (0x50 0x4E 0x47 for PNG images, 0x25 0x50 0x44 0x46 0x2D for PDF documents) or representing the initials of the creator of ZIP format (0x50 0x4B for Phil Katz). Searching magic numbers is far more reliable but more time consuming than checking a given file extension. A magic number is part of the file structure. It is checked and validated at the very beginning by the parsing libraries used to decode the file content. Affecting the signature, i.e. changing the magic number, alters the internal structure of the file and makes it not readable anymore. Thus, altering an exogenous metadata is easier and less risky than altering the magic number. People interested in hiding information will not choose to alter the file content, they will change the associated metadata for a shallow camouflage, or use encryption techniques or steganographic techniques for a deep camouflage. When a disk partition is corrupted and led to the loss of metadata such as directory folders, recovery tools use magic numbers to recover file types and set appropriate filename extensions to the recovered files. While there is no guarantee of the absence of conflict between a valid magic number and an arbitrary sequence of bytes, one may consider that using magic numbers is a good tradeoff to reliably detect file types. Checking for the presence of a short sequence of bytes at the beginning of the file is more expensive than checking a filename extension, but far more reliable. It would also be too expensive to parse the whole

file with the appropriate parsing library to check for a possible collision.

Today, with the prevalence of statistical approaches and deep learning techniques, most research focuses on content-based analysis [24, 25, 26]. Most works are oriented towards the type identification of file fragments and corrupted files. The binary content is vectorized: an associated n-gram vector representation is computed, with n typically ranging from 1 to 2. Unigram or 1-g ($n = 1$) corresponds to the Byte Frequency Distribution (BFD). The n-gram vector of a targeted segment is compared to the reference n-gram vectors which are computed for each file type on a training set. Identifying the file type consists in finding the closest reference profile. It can be achieved with distance metrics such as the cosine distance, with machine learning models, such as SVM, k-NN, or with deep learning techniques where, for instance, the input layer encodes each dimension of the binary vector and the output layer corresponds to each file type.

Table 1: Literature review on file type identification methods (scientific papers and software), content extended from [25].

Paper	**Year**	**Principle**	**#types**	**Dataset**	**Accuracy**
Sester et al. [25]	2021	SVM linear kernel	6	6000	91.4%
Al Neaimi et al. [27]	2020	deep learning	8	4000	99%
Karampidis et al. [28]	2018	byte frequency distribution + neural network	4	2200	97.6%
Beebe et al. [26]	2016	K-Means, Hierarchical classification	50	2600	74.1%
Evensen [29]	2015	n-gram analysis with naive Bayes classifier	6	60000	99.5%
Amirani et al. [30]	2013	PCA + Neural Networks feature extraction, SVM	6	1200	99.1%
Cao et al. [31]	2010	Gram Frequency Distribution	4	1000	90.3%
Dhanalakshmi & Chellappan [32]	2009	Feature Selection +KNN Classifier	10	5000	90.5%

Software	**Year**	**Principle**	**#types**	**Language**	**Licence**
File (Unix)	1973	magic number +language	338	C	open-source
Fidentify [19]	2012	magic numbers +binary	440	C	open-source
ForENSIque	2021	magic numbers	155	Rust	open-source
Guess-file-type [18]	2018	magic numbers +extension	985	Javascript	open-source
detect-file-type [17]	2016	magic numbers	94	Javascript	open-source
filetype [16]	2016	magic numbers	64	Python	open-source
EnCase [21]	2015	magic numbers	406	EnScript/C++	commercial
file-type [15]	2014	magic numbers	135	Javascript	open-source
Autopsy [22]	2012	magic numbers +extension	1524	Java	open-source
TrID [20]	2003	magic numbers	14374	Python	commercial

Table 1 lists most research works and software in the literature for file type identification. We can draw the following conclusions. All scientific papers on file type identification are based on machine learning [25]. The number of file types (or classes in this case) is quite low since the approach does not scale very well. The work done by [26] considered 50 file types (the highest number) but the accuracy is quite low (74.1%). Last, all these works consider low dataset sizes (the largest one contains 60000 files). For all these reasons, these approaches are limited for an operational use for a forensic expert, and are not really used in practice. Many software solutions are available, some are commercial or open-source on GitHub. We can notice that the number of file types considered in each application is very different but much higher than in scientific papers. Most of these solutions use magic numbers and sometimes

other information. One of the problems is that their accuracy and computation efficiency are not available.

In this paper, we realize a comparative study of file type identification solutions. Scientific papers are not considered as they consider too few numbers of file types to be used in an operational context. We consider in this work all software solutions listed in Table 1.

3. Comparative study

In order to compare different tools for file type identification, we need some datasets, evaluation metrics and a benchmarking platform.

3.1. Datasets

In this work, we use two datasets corresponding to different testing scenarios, different number of files, file types and file sizes.

1. Dataset 17500: It is composed of 17,500 files with 110 different file types, it consists of various documents retrieved from public databases [23], images found on the internet and files generated with converters. Figure 1 describes this dataset.

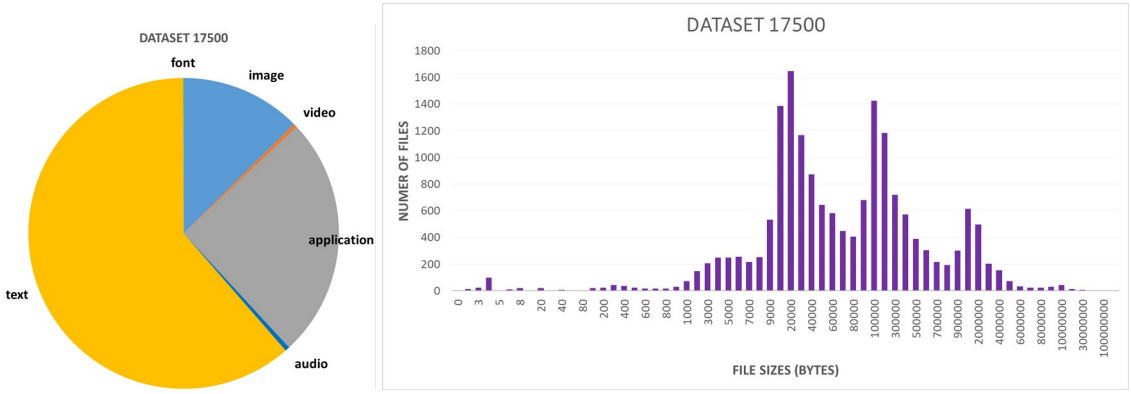


Figure 1: Dataset 17500: Distribution of file size and document types.

2. Dataset 1M: The second corpus is composed 986,263 files with 63 different file types from [23]. It is characterized in Figure 2. This dataset is used to estimate performance at scale (much more files and fewer file types) to simulate operational conditions.

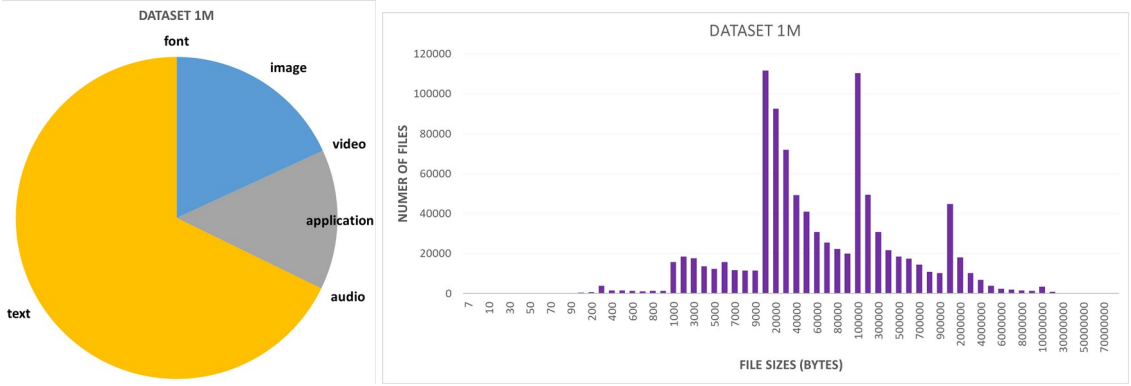


Figure 2: Dataset 1M: Distribution of file size and document types.

Concerning the ground truth (expected file type), we consider that all file types are correct (dataset used in the literature). However, we have made random manual checking to ensure confidence in the quality of this dataset.

3.2. Tested tools

Many tools have been designed for file type identification: some of them were initially implemented for Digital Forensics purposes, some others were designed for other purposes such as checking the file type to prevent users from unintentionally uploading files on a server with an incorrect type. For the evaluation, we considered (see Table 1):

- Unix command File as baseline solution,
- Four starred projects on GitHub: filetype (311 stars), file-type (2.2k stars), detect-file-type (11 stars), guess-file-type (4 stars).
- Four well-known tools in digital forensics investigations: fidentify , which is a module of the two-in-one TestDisk & Photorec suite, TrID, an extensible module with no hard-coded rules, and the widely used software Autopsy and EnCase
- A last tool provided by the academic community for evaluation: ForENSIque

We present each tool in the following.

3.2.1. Fidentify

Fidentify [19] is a well-known forensic tool developed by Christophe Grenier since 2011. It uses the Photorec signature database to determine the file type. Fidentify can also analyze a whole directory. 440 file types are considered in this application.

3.2.2. TrID - File Identifier

TrID [20] is a well-known forensic tool developed par Marco Pontello since 2003. TrID exploits a definition database describing recurring patterns for supported file types. The definition database evolves rapidly, it actually contains 14374 file types but grows fast.

3.2.3. Filetype Python Module

Filetype [16] is a python module defined in 2016. When given the path to a file, it returns an object containing the estimated file type and the MIME type, using magic numbers. Only 64 file types are considered in this application.

3.2.4. File-type Javascript Package

File-type [15] is a JavaScript package which detects file types by checking signatures (i.e. magic numbers). This package handles binary-based file formats, not text-based formats. File-type takes as input the path of a file and returns the file type as well as its MIME type. It can also determine this information by taking a stream as input, for instance a request to a server, or even the content of a buffer. This module works asynchronously, which makes it possible not to block other tasks which are executed simultaneously. 135 file types are handled by this application.

3.2.5. Detect-file-type Javascript API

Detect-file-type [17] is a JavaScript API based on File-type. Contrary to File-type , it handles not only binary formats, but also textual documents (e.g. xml, svg, html). It returns an object containing the MIME type and the assumed extension of the entry. Detect-file-type offers the user to add their own signatures and add their own file format detection methods. 94 file types are recognized by this application.

3.2.6. Guess-file-type Javascript API

Guess-file-type [18] works on a file passed as input. It carries out several tests to try to determine as precisely as possible what the type of the file is. When it succeeds, the module returns only a MIME type, it is possible to use a function to deduce the extension from the MIME type. Guess-file-type offers the possibility to use the tests separately. A detection by signature (i.e. magic number) is available: 34 file types are handled. The attribution of a MIME type from the file type is also proposed: 985 file types are mapped to 766 default MIME types.

3.2.7. ForENSIque Filetype Rust Module

ForENSIque is a software created by ENSICAEN students in France. The code is not publicly available, but was kindly provided by the authors for testing. It has been developed in Rust language and uses magic numbers to recognize 155 file types through internal mapping. The first 16 bytes of the file are retrieved then analyzed. Two cases arise, either the first four bytes are one of the two predefined sequences, 'FORM' and 'RIFF', or not. In the first case, depending on whether it is 'FORM' or 'RIFF', the four bytes 8 to 11, are considered. That method allows the application to recognize 20 file types such as IFFCelAnimation (.anim) or AudioVideoInterleave (.avi). In the second case, the internal mapping is used. From 2 to 16 bytes long, 134 magic numbers are mapped to the corresponding file types. At this point, up to 154 file types can be handled. If the file type is not found, a last test allows checking the bytes further to detect tar archives. The output displays the time statistics and the number of files processed. The processing results, including the file types detected, are stored in a database.

3.2.8. EnCase Endpoint Investigator software

EnCase[21] is a software created by Shawn H. McCreight for the company Guidance Software, later acquired by OpenText. We had the opportunity to use a trial version of EnCase Endpoint Investigator which is one of their products. The feature that interested us analyzes the signature of the file and if it corresponds to a known one, then it indicates if the detected file type is consistent with the one suggested by the extension. If not, it returns a file type according to the shown extension.

3.2.9. Autopsy Digital Forensics software

Autopsy 4.19.3[22] is a widely used graphical user interface (GUI) that operates "the sleuth kit" to analyze files from disk images and is maintained by Basis Technology Corp. When the file signature exists, this software analyzes it to predict its corresponding MIME type and if it is not known, it may rely on shown extension.

3.2.10. Unix file command

file command in Unix and Unix-like operating system allows file type detection by performing several tests, including position-sensitive tests using a textual database of magic numbers, located in the magic file.

3.3. Performance metrics

In this study, we used two metrics to evaluate the performance of each tool by considering the following metrics:

1. Computation time: we measure the time needed for the processing (on the same computer). As this kind of tool may be used on a hard disk with millions of files, this is a key relative indicator.
2. Accuracy: we compute the file type recognition accuracy. The two tested datasets used in this work provide the ground truth. In some test scenarios, we modify the file extension or suppress it to analyze the impact on accuracy.

3.4. Benchmarking platform

In order to benchmark tools that could be used in Digital Forensics, we design a software platform called G'DIP for GREYC Digital Investigation platform (see Figure 3). This platform allows us to process data from any sources (device, computer, hard drive, file directory . . .). A python core applies any filter (like file type identification) to all files present in the data source. The results are stored in a database that can be processed by a graphical user interface. Statistics about the datasets, represented by diagrams, and evaluation results can be visualized by the user. The main objectives of this platform are 1) to benchmark tools for digital investigation, 2) to provide a software tool for teaching digital investigation and 3) to propose an operational and open-source tool for investigators on real data. This platform has been used to generate results in this article.

Figure 4 focuses on the different steps for benchmarking a digital investigation tool. For this kind of activity, a reliable ground truth is necessary. A mapping function is necessary because the outputs of all tested tools are not necessarily the same. All benchmarking results are saved in the processed databases. We applied this process to file type identification tools on the previous datasets and we compute the associated performance metrics for visualizing experimental results.

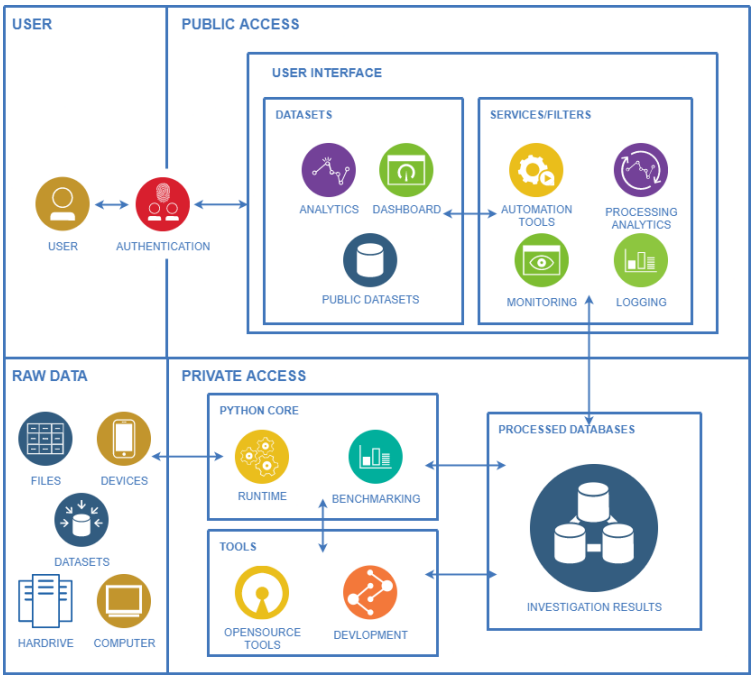


Figure 3: The G'DIP software platform used in this work for evaluating investigation tools.

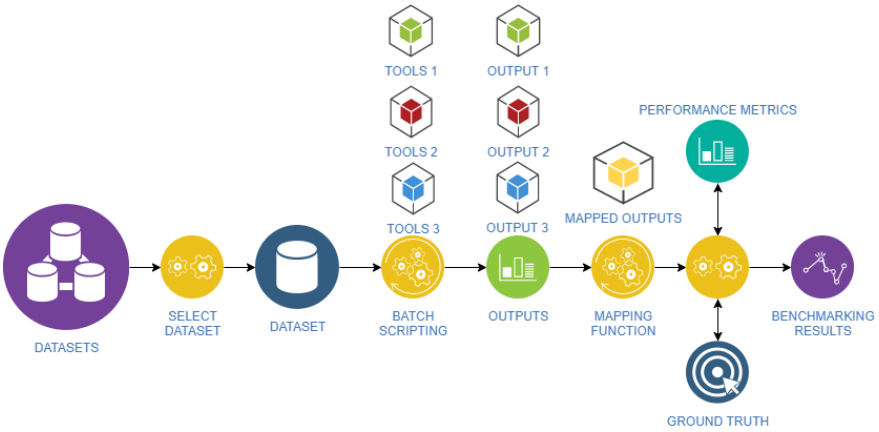


Figure 4: Different processes for benchmarking investigation tools on datasets with ground truth.

4. Experimental results

In this section, we present the experimental results we obtained on the 2 datasets for the 10 tested tools. We structured experimental results by the questions we wanted to answer on file type identification.

4.1. What is the file type identification performance of tested tools?

We first consider the ability of the tested tools to correctly identify the file type. After running tests with the 10 tools, we have gathered obtained results in Tables 2 and 3. Table 2 shows the accuracy values of each tool and Table 3 shows the associated computation time. We analyze these results for each dataset.

Table 2: Accuracy of file type identification for each tested tools on the two datasets.

Tools	17500	1M
filetype	36.2%	43.3%
file-type	49%	65.8%
detect-file-type	74.6%	80%
guess-file-type	34.1%	38.2%
ForENSIque	59.3%	65.8%
Fidentify	**94.4%**	**98.1%**
TrID	74.9%	83.9%
EnCase	77.3%	82.1%
Autopsy	90.6%	88.6%
file	84.7%	85,8%

Table 3: Computation time on the two datasets.

Tools	17500	1M
filetype	5m8s	4h40m15s
file-type	40s	5h32m18s
detect-file-type	1m8s	3h44m42s
guess-file-type	5m3s	4h51m24s
ForENSIque	**30s**	6h40m53s
Fidentify	1m23s	6h04m47s
TrID	21m9s	17h55m15s
EnCase	44s	4h21m29s
Autopsy	6m35s	9h39m40s
file	2m09s	**1h09m47s**

4.1.1. Dataset 17500

On this dataset, the fastest tool is ForENSIque with 59.3% of correct identification while Fidentify , with 94.4% is the best in terms of identification, followed by Autopsy with 90.6% and unix file command with 84.7% . With lower identification rate, EnCase and detect-file-type are fast however. Which TrID is not, with more than 20 minutes of computation time. Lastly, we can see that guess-file-type is the less efficient tool with only 34.1% of identification and more than 5 minutes in computation time.

4.1.2. Dataset 1M

As it has been said, this dataset is meant to be a more realistic case with fewer challenging file types. We notice an improvement in terms of identification, observable for all tools except for Autopsy Apart from that, there is no major change in their performances. Unix command File is the fastest while achieving 85.8% of correct identification, with Fidentify that is still the best tool in this field, followed by Autopsy TrID and EnCase

4.1.3. Discussion

Thanks to those results, we can draw some conclusions. First, filetype (python), file-type (node.js), guess-file-type , ForENSIque, and TrID are not sufficiently efficient. The first four because of the lack of performance in retrieving the format of files and the last two because of too high an execution time. detect-file-type and EnCase are very similar in terms of performance and while quite fast, they are not accurate enough to be kept. Finally, there are only three modules left: Autopsy File and Fidentify . Each is worth being considered, however, Autopsy and File are not as accurate as Fidentify .

4.2. Do existing tools use file extensions for their file type identification?

In this experiment, we study how existing tools use the file extension to identify its type. To test that, we removed all file extensions in all datasets. As we can see in Table 4 describing results for the two datasets, all tested tools

Table 4: Influence of File type Accuracy (Datasets 17500 and 1M).

	Dataset 17500			**Dataset 1M**		
tools	with ext	ext removed	renamed ext	with ext	ext removed	renamed ext
filetype	36.2%	36.2%	36.2%	43.3%	43.3%	43.3%
file-type	49%	49%	49%	65.8%	65.8%	65.8%
detect-file-type	74.6%	74.6%	74.6%	80%	80%	80%
guess-file-type	**94.8%**	34.1% (-60,7%)	35% (-59,8%)	96.6%	38.2% (-58,4%)	38.2% (-58,4%)
ForENSIque	59.3%	59.3%	59.3%	65.8%	65.8%	65.8%
Fidentify	94.4%	**94.4%**	**94.4%**	98.1%	**98.1%**	**98.1%**
TrID	74.9%	74.9%	74.9%	83.9%	83.9%	83.9%
EnCase	91%	77.3% (-13,7%)	77.3% (-13,7%)	**98.2%**	82.1% (-16,1%)	82.1% (-16,1%)
Autopsy	92.3%	90.6% (-1.7%)	90.6% (-1.7%)	93.1%	88.6% (-4.5%)	88.6% (-4.5%)
File	84.7%	84.7%	84.7%	85.8%	85.8%	85.8%

obtain the exact accuracy when extensions have been removed than in the case where extensions were present except for EnCase, Autopsy and guess-file-type . We observe this last tool having an important decrease in performance. The obtained results are not surprising as most tools use magic numbers to detect file types. This experiment shows that guess-file-type is not a good choice for digital investigation purposes (where file extensions could have been voluntarily altered). It is also surprising to have a decrease of performance for the EnCase commercial solution when file extensions have been removed.

4.3. Is there any impact on the performance of an attack consisting in altering the file extension?

In a similar way, we altered in this case all file extensions present in all datasets (see Table 4). Once again, almost all tested tools obtain the exact accuracy than in the case when correct extensions were present, except once again for Autopsy EnCase and guess-file-type .

4.4. Is there any difference of precision between existing tools?

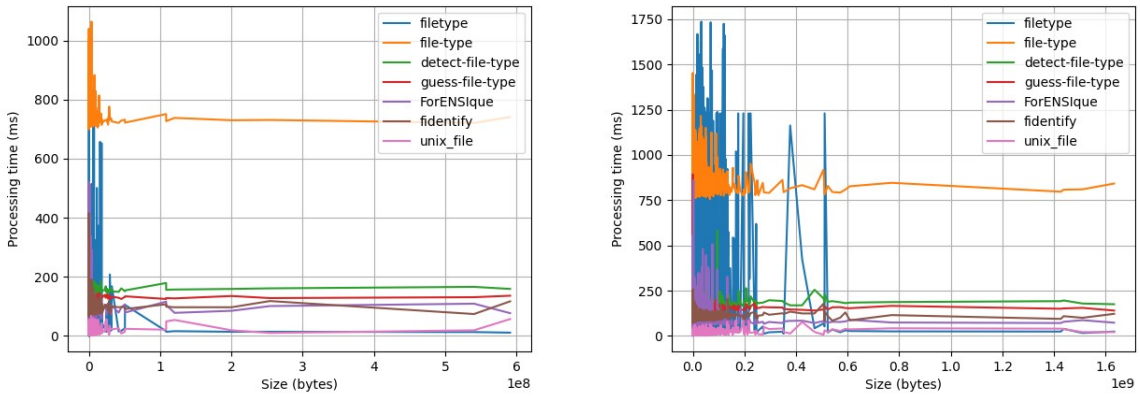
When we tested the different tools from the literature, we observed many differences in their outputs. First, some tools return more precise outputs than others. We show in Table 5 the outputs of the tested tools for 4 popular extensions: ods, odt, pps, ppt. While some tools identify these files as zipped archives, some others identify them as presentation and spreadsheet documents. In fact, the zip archive file type is used to store the different objects of these documents. To determine if the file type is correct previously, we choose to adopt a coarse grain classification. We could have weighted results considering the precision of the outputs using ontology-based metrics [33, 34]. It shows that MIME type is not a shared output standard: some tools use ad hoc string to name file types, and sometimes there is no existing MIME type of very specific extensions. In order to handle this problem, we had to define our own nomenclature and we had to define mapping functions to compare the different tools (see Figure 4).

Table 5: Examples of outputs of tools for 4 File types.

ods	**odt**	**pps**	**ppt**
application/zip	application/zip	application/x-msi	application/x-msi
application/zip	application/vnd.oasis.opendocument.text	application/vnd.ms-powerpoint	application/vnd.ms-powerpoint
ZipArchive	ZipArchive	CompoundFileBinaryFormat	CompoundFileBinaryFormat
application/vnd.oasis.opendocument.spreadsheet	application/vnd.oasis.opendocument.text	application/msword	application/msword
application/vnd.oasis.opendocument.spreadsheet	application/vnd.oasis.opendocument.text	application/vnd.ms-powerpoint	application/vnd.ms-powerpoint

4.5. Is there any relationship between the file size and computation time of existing tools?

This experiment consists in running existing tools on each file in order to obtain the computation time and the associated file size. Since the two software EnCase and Autopsy are not suitable for this type of process, we did not include them in the study (they process files in a directory in a batch mode). We plot the file size versus the computation time in order to identify a potential correlation. We experimented on the two datasets to get a better representation, then the results were ordered by size for each dataset, which leads to two diagrams (see Figure 5). The graphs tend to prove that the computation time does not depend on the file size. It tends to be constant after 300 kb. We make the hypothesis that the computation time is linked to the size of the signature database, the search algorithm of a matching signature, and the programming language.



(a) Dataset 17500

(b) Dataset 1M

Figure 5: Analysis of the processing time versus the file size for the two datasets.

4.6. Which formats are not recognized by existing tools?

In this part, we study the file formats that are not correctly recognized. As for illustration, we consider two tools namely Fidentify and EnCase. We list for all datasets, extensions that are not recognized by these two tools:

- Fidentify :
  7z, 8svx, aac, acbm, amv, anim, avro, cin, crx, deep, dmg, dpx, dwf, faxx, fbx, flif, fm, glb, mmf, parquet, tga, wbmp.
- EnCase
  avro, cin, crx, dwf, fbx, flif, glb, mmf, parquet.

When looking at these lists, most extensions are not well known or may correspond to very different contents. Thanks to a web service called files.tips[1], we can have some information on some of these extensions:

- 7z: 7-Zip Compressed Format format;
- dpx: Digital Picture Exchange Format format;

According to files.tips, many of these extensions are not popular. Figure 6 gives the popularity value for each of these extensions by files.tips. None information is given on how it is computed. It is probably related to the number of requests to open all these file types (this service helps users to open any kind of file). Unrecognized extensions by Fidentify have on average a popularity score equal to 2.7 out of a maximum score of 5 where the ones not recognized by EnCase are on average 2.5. A file extension with 0 as popularity is not even known by the files.tips service. It appears clear that some popular extensions such as '7z' or 'dmg' are present in these lists and should be considered as important to be recognized.

[1] https://files.tips/

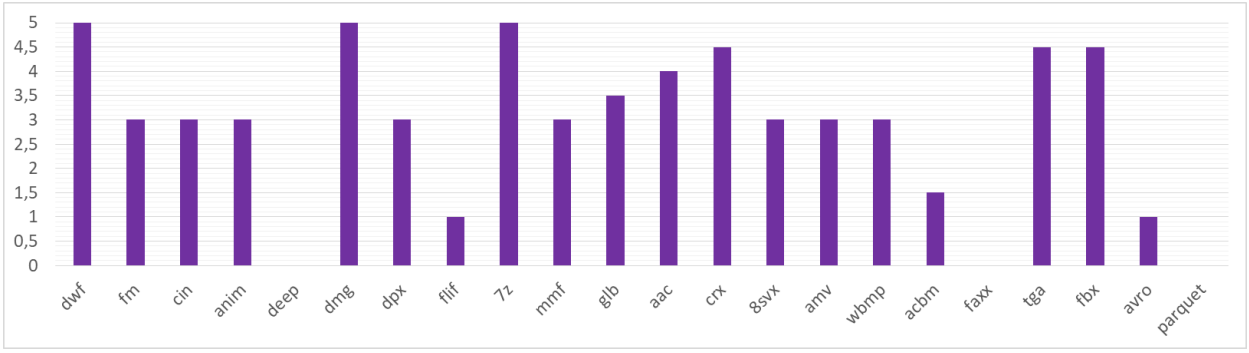


Figure 6: Popularity of file extensions (evaluated by files.tips) that are not recognized by Fidentify or EnCase.

4.7. Can we combine existing tools to enhance their performance?

Instead of looking for a perfect tool, it could be more efficient to combine multiple ones. In this work, we did not consider all the combinations of the 10 tested tools. As an illustration, we propose some combinations, starting with both Fidentify and ForENSIque tools. Fidentify provided the best result and ForENSIque has been designed by some of our students. The prototype is developed in Python. It takes as input a directory path and analyze all files in the directory. The strategy is to first apply Fidentify on every file since it is the most efficient module, and to use ForENSIque as a fallback when Fidentify fails to identify the type. Algorithm 1 details the combined method we implemented called GreycFiletype.

Algorithm 1 Combining Fidentify and ForENSIque: GreycFiletype

Input
$D = \{f_1, \ldots, f_n\}$: directory with n files $f_i, i = 1 : n$,
Output $A = \{E_1, \ldots, E_n\}$: Extension of each file $f_i, i = 1 : n$ or unknown.
for $i = 1, \ldots, n$, **do**
    Compute E_i = Fidentify(f_i).
    if E_i is Unknown **then**
        Compute E_i = ForENSIque(f_i).

Following the same principle, we have combined Fidentify with Autopsy and TrID. In order to determine the performance of these new tools, we tested them with the same evaluation protocol. The results are shown in Tables 6 and 7. The combination of Fidentify and ForENSIque demonstrates a significant improvement in terms of accuracy. The computation time is increased compared to Fidentify which can be explained by the naive implementation of this proof of concept. Conversely, the increase due to the other two combinations is less, if any. These results show that Fidentify can be improved but that it is difficult to do so, particularly on a dataset with few uncommon file types.

Table 6: Accuracy of the detection of the file format.

	Original modules				**Combined modules**		
Datasets	Fidentify	ForENSIque	Autopsy	TrID	Fid + For	Fid + Aut	Fid + TrID
17500	94.4%	59.3%	90.6%	74.9%	**96.2%** (+1.8%)	95.3% (+0.9%)	94.9 (+0.3%)
1M	98.1%	65.8%	88.6%	83.9%	**98.3%** (+0.2%)	98.1%	98.1%

4.8. Summary

In Figure 7, we show the cumulated accuracy on all datasets for each tool. We can draw the following conclusions:

Table 7: Execution time

	Original modules				Combined modules		
Datasets	Fidentify	ForENSIque	Autopsy	TrID	Fid + For	Fid + Aut	Fid + TrID
17500	1m23s	30s	6m35s	21m09s	1m42s	2m33s	5m8s
1M	6h04m47s	6h40m53s	9h39m40s	17h55m15s	10h28min36s	12h43m08s	15h08m53s

- filetype , file-type , detect-file-type , File and ForENSIque tools provide globally insufficient performance and should not be used alone.
- guess-file-type , TrID and Encase tools provide good results in general but in case of changed extension (possible attack), the accuracy decreases (a lot for guess-file-type),
- File is clearly the quickest solution with a good performance,
- Fidentify and GreycFiletype provide very good results and are robust to a modification of file extensions.

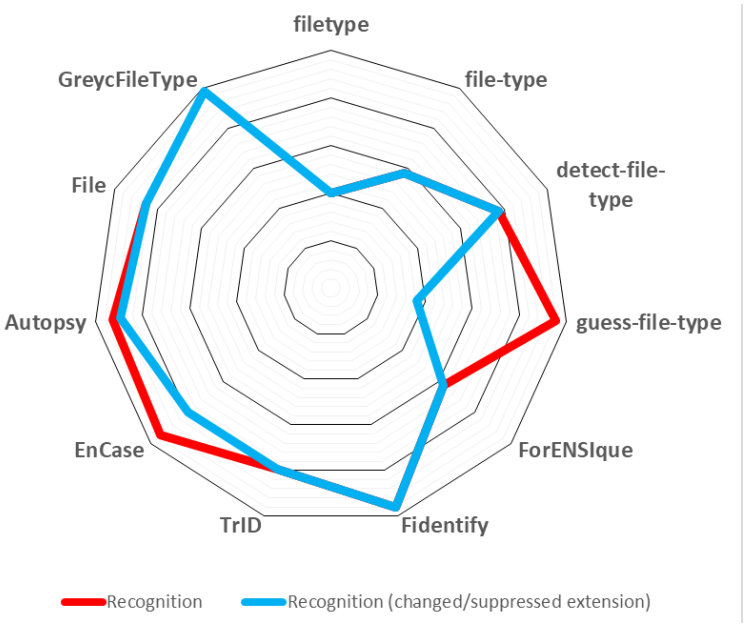


Figure 7: Performance of file type identification by the tested tools in different contexts (expressed by percentage): 1) file type recognition, 2) file type recognition when file extensions have been changed or suppressed.

5. Conclusion and perspectives

In Digital Forensics, investigators have to mine data storage medium in order to recover information or to assess evidence. Open source investigation tools are available and can be very useful to handle cases. However, they mostly lack rigorous quality evaluation. Starred projects and popular tools should be rigorously evaluated to determine whether they can be used in a forensic context, or not. In this study, we make a literature review of existing tools for file type identification. We consider both academic and non-academic forensic tools, including commercial tools used in digital investigation cases. We proposed an objective protocol involving significant complementary datasets and performance metrics for the comparative study of the file identification tools. Two scenarios were considered to

conduct the benchmarking: the first scenario aims to assess the coverage and the accuracy of the tools in terms of file extension detection, the second scenario aims to simulate a realistic set investigators could have to handle in a real case: the number of file types is smaller but the number of files permits to better evaluate speed.

This benchmark shows that Fidentify is the most efficient solution. However, we found that many file formats are not considered by existing tools. All tested tools have a similar behavior concerning their computation time: it is not related to the file size. TrId and EnCase may also be considered as good alternatives concerning accuracy detection. Both tools can be extended with new custom signatures. Concerning TrId, new signatures can be shared and benefit to the community. However, our experiments tend to show that it still remains a very slow tool. The comparative evaluation tends to prove that the three file type identification tools designed for investigation purposes are more accurate than open source repos available on GitHub, even well-rated ones. Finally, we showed that the combined tool, called GreycFiletype, can improve the accuracy of file type identification (near 98%).

As perspectives, we plan to open the used datasets and codes for the research community to stimulate studies in this area. We also plan to consider the feasibility of merging the improvements in Fidentify and Photorec. In order to enhance accuracy results of file format identification, we intend to apply some machine learning techniques by combining different file features such as MIME type, magic numbers, file size, or entropy.

References

[1] L. Caviglione, S. Wendzel, W. Mazurczyk, The future of digital forensics: Challenges and the road ahead, IEEE Security & Privacy 15 (06) (2017) 12–17. doi:10.1109/MSP.2017.4251117.

[2] M. Kirschenbaum, R. Ovenden, G. Redwine, R. Donahue, Digital forensics and born-digital content in cultural heritage collections.

[3] D. Dietrich, F. Adelstein, Archival science, digital forensics, and new media art, Digital Investigation 14 (2015) S137–S145.

[4] J. Jarlbrink, How to approach hard drives as cultural heritage.

[5] F. Flandrin, W. Buchanan, R. Macfarlane, B. Ramsay, A. Smales, Evaluating digital forensic tools (dfts), in: 7th International Conference on Cybercrime Forensics Education and Training CFET 2014, 2014.doi:10.13140/2.1.3293.6004.

[6] W. A. Bhat, A. AlZahrani, M. A. Wani, Can computer forensic tools be trusted in digital investigations?, Science & Justice 61 (2) (2021) 198–203. doi:https://doi.org/10.1016/j.scijus.2020.10.002.
URL https://www.sciencedirect.com/science/article/pii/S1355030620303002

[7] I. T. Lazaridis, S. Pouros, T. Arampatzis, Evaluation of digital forensics tools on data recovery and analysis, in: The third international conference on computer science, computer engineering, and social media (CSCESM2016), 2016, pp. 67–71.

[8] G. Horsman, Tool testing and reliability issues in the field of digital forensics, Digital Investigation 28 (2019) 163–175. doi:https://doi.org/10.1016/j.diin.2019.01.009.
URL https://www.sciencedirect.com/science/article/pii/S1742287618303062

[9] E. Casey, The chequered past and risky future of digital forensics, Australian Journal of Forensic Sciences 51 (6) (2019) 649–664. arXiv:https://doi.org/10.1080/00450618.2018.1554090, doi:10.1080/00450618.2018.1554090.
URL https://doi.org/10.1080/00450618.2018.1554090

[10] P. M. Dimpe, O. P. Kogeda, A model for evaluating digital forensic tools, Journal of Engineering and Applied Sciences 14 (2019) 7048–7058. doi:10.36478/jeasci.2019.7048.7058.
URL https://medwelljournals.com/abstract/?doi=jeasci.2019.7048.7058

[11] J. Beckett, J. Slay, Digital forensics: Validation and verification in a dynamic work environment, in: 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07), IEEE, 2007, pp. 266a–266a.

[12] K. Sindhu, B. Meshram, Digital forensics and cyber crime datamining, Journal of Information Security 3 (3) (2012) 196–201. doi:10.4236/jis.2012.33024.

[13] H. Arshad, A. B. Jantan, O. I. Abiodun, Digital forensics: review of issues in scientific validation of digital evidence, Journal of Information Processing Systems 14 (2) (2018) 346–376.

[14] L. Aronson, J. Van Den Bos, Towards an engineering approach to file carver construction, in: 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, IEEE, 2011, pp. 368–373.

[15] S. Sorhus, File-type javascript api (2014).
URL https://github.com/sindresorhus/file-type

[16] h2non, Filetype python api (2016).
URL https://github.com/h2non/filetype.py

[17] D. Paloskin, detect-file-type javascript api (2016).
URL https://github.com/dimapaloskin/detect-file-type

[18] Kosinix, Guess file type javascript api (2018).
URL https://github.com/kosinix/guess-file-type

[19] C. Grenier, Fidentify: Determine file type using photorec database (2019).
URL https://www.mankier.com/8/fidentify

[20] M. Pontello, Trid - file identifier (2003).
URL https://mark0.net/soft-trid-e.html
[21] G. S. Shawn H. McCreight (1998). [link].
URL https://security.opentext.com/encase-endpoint-security
[22] B. T. Corp (2012). [link].
URL https://www.autopsy.com/
[23] S. Garfinkel, P. Farrell, V. Roussev, G. Dinolt, Bringing science to digital forensics with standardized forensic corpora, digital investigation 6 (2009) S2–S11.
[24] S. Gopal, Y. Yang, K. Salomatin, J. Carbonell, Statistical learning for file-type identification, in: 2011 10th International Conference on Machine Learning and Applications and Workshops, Vol. 1, 2011, pp. 68–73. doi:10.1109/ICMLA.2011.135.
[25] J. Sester, D. Hayes, M. Scanlon, N.-A. Le-Khac, A comparative study of support vector machine and neural networks for file type identification using n-gram analysis, Forensic Science International: Digital Investigation 36 (2021) 301121.
[26] N. Beebe, L. Liu, M. Sun, Data type classification: Hierarchical class-to-type modeling, in: IFIP International Conference on Digital Forensics, Springer, 2016, pp. 325–343.
[27] M. Al Neaimi, H. Al Hamadi, C. Y. Yeun, M. J. Zemerly, Digital forensic analysis of files using deep learning, in: 2020 3rd International Conference on Signal Processing and Information Security (ICSPIS), IEEE, 2020, pp. 1–4.
[28] K. Karampidis, I. Deligiannis, G. Papadourakis, Combining genetic algorithms and neural networks for file forgery, Machine Learning Paradigms: Advances in Data Analytics 149 (2018) 317.
[29] J. D. Evensen, Clustered file type identification, Master's thesis, Universitetet i Agder; University of Agder (2015).
[30] M. C. Amirani, M. Toorani, S. Mihandoost, Feature-based type identification of file fragments, Security and Communication Networks 6 (1) (2013) 115–128.
[31] D. Cao, J. Luo, M. Yin, H. Yang, Feature selection based file type identification algorithm, in: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, Vol. 3, IEEE, 2010, pp. 58–62.
[32] R. Dhanalakshmi, C. Chellappan, File format identification and information extraction, in: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009, pp. 1497–1501.
[33] D. Maynard, W. Peters, Y. Li, Metrics for evaluation of ontology-based information extraction, in: D. Vrandecic, M. C. Suárez-Figueroa, A. Gangemi, Y. Sure (Eds.), Proceedings of 4th International EON Workshop 2006 Evaluation of Ontologies for the Web Co-located with the WWW2006 Edinburgh, UK, May 22, 2006, Vol. 179 of CEUR Workshop Proceedings, CEUR-WS.org, 2006.
URL http://ceur-ws.org/Vol-179/eon2006maynardetal.pdf
[34] A. Kumar, Govind, M. Spaniol, Semantic search via entity-types: The semannorex framework, in: Companion Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 690–694. doi:10.1145/3442442.3458607.
URL https://doi.org/10.1145/3442442.3458607