Understanding the effects of removing common blocks on Approximate Matching scores under different scenarios for digital forensic investigations

Vitor Hugo Galhardo Moia[1], **Frank Breitinger**[2], **Marco Aurélio Amaral Henriques**[1]

[1]Department of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computer Engineering (FEEC)
University of Campinas, Av. Albert Einstein 400, Campinas SP, 13083-852, Brazil

[2]Cyber Forensics Research and Education Group (UNHcFREG)
Tagliatela College of Engineering, ECECS
University of New Haven, 300 Boston Post Rd., West Haven CT, 06516, USA

[vhgmoia,marco]@dca.fee.unicamp.br, FBreitinger@newhaven.edu

Abstract. *Finding similarity in digital forensics investigations can be assisted with the use of Approximate Matching (AM) functions. These algorithms create small and compact representations of objects (similar to hashes) which can be compared to identify similarity. However, often results are biased due to common blocks (data structures found in many different files regardless of content). In this paper, we evaluate the precision and recall metrics for AM functions when removing common blocks. In detail, we analyze how the similarity score changes and impacts different investigation scenarios. Results show that many irrelevant matches can be filtered out and that a new interpretation of the score allows a better similarity detection.*

1. Introduction

Approximate Matching (AM) algorithms follow the concept of cryptographic hash functions: They allow to create small and compact representations for objects (a.k.a. *digests*). However, they differ from traditional hashing in the sense that small changes in an object results in a small change in the digest. Thus, instead of comparing objects, one can compare two digests in order to identify similarity. This paper focuses on bytewise AM which operates on the byte-level and is used during digital forensics investigations [Breitinger et al. 2014]: One application for such functions is known-file identification during a digital forensic investigation, i.e., an investigator can identify modified versions of documents, embedded objects, artifacts in network packets, or do malware clustering.

The problem of current AM functions is that too many matches with irrelevant results (i.e., many false positives) are produced due to common structures found in objects which was first addressed by [Foster 2012] who called them *common blocks*. Examples of such common data are header/footer information, color palettes, font specifications, or other data structures belonging to particular software vendors.

We extend the work of [Moia et al. 2019] who suggested removing the common data from the similarity digest when using AM functions to assess objects similarity. Their results showed significant improvements. In this paper, we evaluate how the removal of the common blocks affects precision/recall rates of AM. We analyze, for different

scenarios, how the similarity detection is impacted and show that many matches that occurred due to application-generated content have their score zeroed out when common blocks are removed. Consequently, the number of matches practitioners has to deal with is significantly reduced. We also analyzed how the score produced by AM changes and that many matches with low scores, recommended to be ignored before, are now of interest.

2. Related work

Several AM functions have been proposed over the last years, where the most prominent ones seem (chronological order): ssdeep [Kornblum 2006], `sdhash` [Roussev 2010], mrsh-v2 [Breitinger and Baier 2013], TLSH [Oliver et al. 2013] and LZJD [Raff and Nicholas 2018]. Since our work utilized `sdhash`, the next section provides some extra details about it.

2.1. `sdhash`

Proposed by [Roussev 2010], `sdhash` aims at extracting statistically improbable features of an object to create its digest. A feature in this context is defined as a sequence of β bytes (64 bytes by default). A sliding window moves through the object byte-by-byte extracting all features, starting at the first object byte. After the feature extraction process, the features with minimal/maximal Shannon entropy value are removed and the remaining ones with the lowest entropy scores are chosen to be hashed (SHA-1) and inserted into bloom filters [Bloom 1970]. Each filter encompasses at most 160 features, and new ones are created every time a filter reaches its capacity. The final object digest is the concatenation of all filters. To compare two `sdhash` digests, one needs to compare the bloom filters against each other using Hamming distances, resulting in a normalized score from 0 (dissimilar) to 100 (identical or very similar).

2.2. Similarity classes

Whenever AM functions provide a score > 0 about a comparison of two objects, the similarity detected can be related to three classes, as defined by [Moia et al. 2019]:

User-generated content (*UGC*): Data created by users, such as text, pictures, tables, etc. This can be the most relevant class of similarity.

Template content (*TC*): Data created by users. It can repeat over many different files. An example is a company's document template. Every file created by this company will contain the same header, footer, and logo picture. This is a form of similarity less relevant but could be useful for practitioners in specific cases. However, it may also lead to many irrelevant results.

Application-generated content (*AGC*): Data created by applications. An example is file-header information with metadata required to access the file. This information is usually shared among (almost) all files of the same type and, in some cases, even with files of different types. This class should be avoided in investigations and hence, is the least relevant to identify object similarity.

2.3. Common blocks

Common blocks are pieces of similar data found in different objects. Since they may repeat in many different files, they are not suitable for assessing similarity in some contexts.

[Foster 2012] was the first to call attention to this matter using a hash-based carving approach. In their work, blocks of fixed-size (512 or 4096 bytes) were hashed and compared with blocks of a particular file. Since many common blocks repeat over different files, it is hard to prove the existence of a given file on a media under analysis only by some pieces of it. By using a database to filter out the blocks that repeated several times, Foster removed them and focused on *distinct blocks* (the ones that occur only once) to perform an analysis. In addition, some blocks present a predictable behavior, and by excluding those with low entropy values (i.e., a block with repetition of the same byte) or having known n-grams, many common blocks were filtered out. [Garfinkel and McCarrin 2015] extended the prior work by proposing additional rules, stating that the entropy calculus was not enough. [Gutierrez-Villarreal 2015] went into a different direction and said that all proposed rules were redundant and can be replaced by a single one. Their experiments showed that blocks (4096-byte) with entropy of 10.9 or higher were adequate to remove many common blocks.

[Moia et al. 2019] replaced the hash-based carving method by the approximate matching functions to extract the common blocks (referred to as common features). Instead of using entropy values to identify the common features, the authors created a database to store the features of a comprehensive dataset[1] with many files of different types, and those features that repeated more than N times, were considered common and removed from digests. The authors showed how frequent some features are and how they are spread across various file types. A significant reduction in the number of (undesirable) matches was observed in their results. Many of these undesirable matches were due to template or application-generated content, the similarity classes least attractive in most digital forensic investigations. In the remaining of this paper, we use the same term (*common features*) to refer to common blocks since they are extracted using an AM function.

3. Research direction, design decisions and implementation

[Roussev 2011] found that `sdhash` matches with scores of < 21 contain many irrelevant results and recommended to ignore them (except for text files where the author recommended ignoring scores < 5). By removing the common features, we expect that most of the similar content of two matched files be related to user-generated content, and even low `sdhash` scores present relevant matches. This way, a new interpretation of the score produced by `sdhash` is necessary.

In contrast to the work of [Moia et al. 2019] that showed how common are the common features, how they are spread over files, and the impact on the number of matches when we remove them, this work focuses on the impact of such action on the similarity score and measures how it affects precision and recall of the algorithms. Besides, the different classes of similarity will impact on investigations in different ways, according to the goal of the search. This paper addresses the following research questions:

- RQ1 How does the removal of common features impact digital forensics investigations for the different classes of similarity?
- RQ2 How are precision/recall rates affected by the removal of common features?
- RQ3 How is the recommended threshold value of 21 for `sdhash` affected by removing common features?

[1]The t5-corpus was utilized which contains 4457 unique files of eight different file types.

3.1. Procedure overview

To answer our research questions and assess how similarity is affected by removing the common features for the different similarity classes, we simulated a digital forensic investigation where a seized media is compared to a database of known files. We used `sdhash` and `NCF_sdhash` (a modified version of `sdhash` that uses a database for removing the common features [Moia et al. 2019]) to compare the two sets (see Sec. 4) against each other. We considered three different scenarios:

Scenario I: We are interested in finding file matches related to *UGC* and/or *TC*. Any match related to these types is considered true positive; otherwise, false positive.
Scenario II: Here, we are only interested in finding *UGC* matches, considered as true positives. Matches related to *TC* and/or *AGC* are considered false positives.
Scenario III: This scenario ignores *TC* matches to remove their impact on investigations. Matches related to *UGC* are true positives, and those of *AGC* are false positives.

To determine the similarity class of a match, we manually investigated all matches reported by either `sdhash` or `NCF_sdhash` (score > 0). To perform manual comparisons, we either used the appropriate software (e.g., MS office, specific web browser etc.) and, in case of binary comparisons, we used `Bless`[2].

When the files of a match had no visual similarity, such as common text, pictures, tables, or other elements created by users, we classified it as application-generated content (*AGC*). For template content (*TC*) matches, files need to have the same layout but differ in their content. An example is two `doc` files from the same company where both have identical font specifications, elements disposition, header/footer with the company information, logo, etc., but the content is different.

3.2. Terms and metrics used for the evaluation

We present here the terms, definitions, and metrics used in this paper. The metrics used for the evaluation are based on those used in the field of information retrieval [Olson and Delen 2008].

Score (s): the score returned by the AM function.
Threshold (t): value used to separate matches from non-matches.
Feature (f): byte sequence extracted from objects to be used in AM functions.
Common feature: f, where $|f| > N$ (i.e., a feature f is considered common if it repeats more than N times across different files in a given corpus).
Match: a comparison between two files where the score $s \geq t$.
True positive (tp): a match of two similar files.
True negative (tn): a non-match of two different files.
False positive (fp): a match of two different files (false match).
False negative (fn): a non-match of two similar files (false non-match).
Precision: the ratio of the number of relevant results retrieved (tp) to the total number of results retrieved ($tp + fp$).
Recall: the ratio of the number of relevant results retrieved (tp) to the total number of relevant results ($tp + fn$).

[2] https://github.com/bwrsandman/Bless (last accessed 2019-15-05)

F_1 **score:** harmonic mean of recall and precision, combining these two metrics into one that better distinguishes good results (close to 1) from bad ones (close to 0).

$$Precision = \frac{tp}{tp + fp} \qquad Recall = \frac{tp}{tp + fn} \qquad F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

3.3. Common feature database and `NCF_sdhash` implementation

We used the same database and `NCF_sdhash` implementation as the one presented by [Moia et al. 2019]. Besides, we adopted the same N values for `NCF_sdhash` as the paper did and included an additional one: $N > 2$. From now on, the following nomenclature is used to refer to the different settings of `NCF_sdhash` with respect to N: low ($N > \{2, 3, 5\}$), mid ($N > \{10, 20\}$), and high ($N > \{50, 100\}$) values. The database details and all tools used in this work can be found in the GitHub page: https://github.com/regras/cbamf.

4. Experimental results

The *t5-corpus*[3] (4457 objects; 1.78 GiB) was utilized for our experiments which is a collection of real-world data composed of various file types. We broke the corpus into two sets: *Known data set* and *Target data set*. The first one was used as the digital forensics investigator database and the second one to simulate a seized media under analysis. The objects of the *Target data set* were compared against the investigator's database to look for similar files. We limited the target set to 100 objects (76.95 MiB) to simplify the manual part of the analysis. For each file type, we randomly selected between 5 and 20 objects. Table 1 summarizes both sets.

Table 1. Number of files per type on both data sets (extracted from *t5-corpus*)

	html	text	pdf	doc	ppt	xls	jpg	gif	$\sum$
Target data set	20	10	20	20	10	10	5	5	100
Known data set	1073	701	1053	513	358	240	357	62	4357

4.1. Ground truth

Measuring precision and recall rates of the algorithms requires to know the similarity class of each comparison. Thus, we manually compared the 507 unique matches of `sdhash` and `NCF_sdhash` to determine the class (*UGC*, *TC*, *AGC*) of the match. Table 2 summarizes our results.

It is important to note that we were not interested in measuring the accuracy or detection capabilities of `sdhash`. Instead, we want to evaluate how the removal of the common features impacted similarity detection based on AM functions.

4.2. Target data set vs. Known data set

Comparing all files from the *Target data set* and *Known data set* required a total of ($4357 * 100 =$) 435,700 comparisons. Table 3 shows the number of matches for `sdhash` and `NCF_sdhash` for different score ranges.

[3] http://roussev.net/t5/t5.html (last accessed 2019-29-05).

Table 2. Number of file matches per similarity class (ground truth)

Similarity class	Number of file matches
User-generated content (*UGC*)	45
Template content (*TC*)	93
Application-generated content (*AGC*)	369

Table 3. Number of file matches by score range using `sdhash` and `NCF_sdhash` for the sets comparison, discarding common features with occurrences $> N$.

Score	sdhash	NCF_sdhash for N						
		2	**3**	**5**	**10**	**20**	**50**	**100**
= 1	92	8	9	18	13	10	19	18
≥ 1	454	78	103	151	171	188	222	265
≥ 10	187	46	75	105	130	143	143	148
≥ 21	131	28	49	69	98	108	111	112
≥ 50	56	9	18	34	40	54	57	57
≥ 90	20	8	8	14	21	20	20	20
= 100	9	3	3	9	9	9	9	9

We can see a significant reduction in the number of matches for `NCF_sdhash`, especially for low N values. The removal of common features reduced the score of many matches; some cases were filtered out completely. For instance, some matches having $s = 100$ for `sdhash` had $s = 0$ for `NCF_sdhash` for $N > \{2, 3\}$ (e.g., `002123.html` vs. `002096.html` a *TC* match). More details are provided in Table 4 in which the removal of common features made s decreases as N got lower. Template content matches are challenging to detect and remove since they depend on the number of files sharing the same layout stored in the reference database. In our experiments, we had only a few instances of each template available, which is the reason why low N values worked well in removing related features and decreasing the similarity score.

Table 4. The impact of common features removal on the score of some file comparisons. All cases reported here are Template Content matches.

Query set file	Known set file	Score (0 - 100) for N							
		sdhash	**2**	**3**	**5**	**10**	**20**	**50**	**100**
`002123.html`	`002096.html`	100	0	0	100	100	100	100	100
`000214.html`	`003083.html`	84	2	17	75	84	84	84	84
`004338.html`	`004509.html`	81	0	0	0	0	81	80	80
`000251.doc`	`002145.doc`	72	0	66	70	71	71	72	72
`003751.html`	`002789.html`	62	0	0	0	46	45	61	62
`000986.ppt`	`003662.ppt`	11	10	10	10	10	10	10	10
`004338.html`	`000918.html`	4	0	0	0	0	4	4	4

Table 5 presents a few examples of matches related to application-generated content. Some matches having high scores reported by `sdhash` had a $s = 0$ for `NCF_sdhash` (e.g., `002394.doc` vs. `004066.doc`); the content shared between the matched files was only related to *AGC*. In other cases, removing common features just reduced s (e.g., `001675.pdf` vs. `000746.pdf`), showing that besides the common structure data, the objects shared some *UGC*. A third case shows that some compar-

isons (e.g., `001675.pdf` vs. `002203.pdf`) had higher scores for `NCF_sdhash` than `sdhash`, even tough the files were visually different (no user-generated content)[4].

Table 5. The impact of common features removal on the score of some file comparisons. All cases reported here are Application-Generated Content matches.

Query set file	Known set file	Score (0 - 100) for N							
		sdhash	2	3	5	10	20	50	100
002394.doc	004066.doc	56	0	0	0	0	0	0	0
003047.pdf	001939.pdf	45	0	0	0	0	0	0	0
001675.pdf	000746.pdf	41	0	21	20	21	20	31	32
000698.doc	004419.doc	38	0	0	0	0	0	0	0
001675.pdf	002203.pdf	24	0	63	66	69	71	55	58
000047.xls	000380.xls	21	21	20	20	22	21	21	21
001239.jpg	002627.jpg	17	0	0	0	0	0	9	13

Removing some undesirable features also made *UGC* features prevail and increase the similarity score, as shown in table 6 (e.g., `003049.pdf` vs. `003046.pdf`). The disposition of the remaining features may have influenced the score[4]. Some matches had about the same scores (e.g., `000380.xls` vs. `000397.xls`).

Table 6. The impact of common features removal on the score of some file comparisons. All cases reported here are User-Generated Content matches.

Query set file	Known set file	Score (0 - 100) for N							
		sdhash	2	3	5	10	20	50	100
002245.html	002238.html	100	100	100	100	100	100	100	100
003299.pdf	003296.pdf	91	95	95	96	98	90	90	90
003049.pdf	003046.pdf	59	92	92	92	93	54	54	54
000380.xls	000397.xls	41	45	41	50	41	41	41	41
001645.doc	001646.doc	23	33	31	32	31	26	24	25
001329.html	001330.html	5	13	13	13	13	13	6	6
004915.html	004914.html	0	18	18	18	18	18	18	18

Remark. [Moia et al. 2019] reported in their experiments that a few scores had minor changes due to hash collisions since `sdhash` uses 160-bit SHA-1 as hash function and `NCF_sdhash` adopts the smaller FNV-1a.

4.3. Impact on similarity score over different scenarios

This section focuses on analyzing matches with $s \leq 21$ to measure the impact on threshold t of removing common features. [Roussev 2011] recommended $t = 21$ to identify relevant matches. After removing the common features, our hypothesis is that even the matches having $s < 21$ will present significant user-generated content since most features related to *TC* and *APG* were excluded.

[4] `sdhash`/`NCF_sdhash` store features into a set of bloom filters (max. of 160 features per filter). The comparison function evaluates the Hamming distances among the filters from each object, selects the maximum value and average all results. We believe that removing some features allowed the matching features to be stored in the same filter (they were more easily stored separately before), increasing s.

Table 7 and 8 show the number of file matches by score (divided by the matching class). For instance, consider $s \geq 15$: `sdhash` had 156 matches, where 30 were *UGC*, 47 *AGC*, and 79 *TC*. Based on these results, we calculated precision, recall, and F_1 score for the three different scenarios, as presented in the following sections.

Table 7. Number of file matches per score and per class - Part I: sdhash and NCF_sdhash ($N > 2, 3, 5$).

Score >=	File Matches in the form: *#matches* (*UGC* - *AGC* - *TC*)			
	sdhash	**N >2**	**N >3**	**N >5**
21	131 (29 - 31 - 71)	28 (18 - 6 - 4)	49 (20 - 11 - 18)	69 (25 - 10 - 34)
20	137 (29 - 36 - 72)	31 (19 - 8 - 4)	53 (21 - 13 - 19)	77 (28 - 13 - 36)
19	138 (29 - 37 - 72)	32 (19 - 9 - 4)	55 (21 - 15 - 19)	79 (28 - 15 - 36)
18	141 (29 - 38 - 74)	33 (20 - 9 - 4)	56 (22 - 15 - 19)	82 (29 - 15 - 38)
17	147 (29 - 43 - 75)	36 (21 - 9 - 6)	60 (23 - 15 - 22)	86 (30 - 15 - 41)
16	151 (29 - 45 - 77)	37 (21 - 9 - 7)	63 (24 - 16 - 23)	88 (31 - 16 - 41)
15	156 (30 - 47 - 79)	39 (21 - 9 - 9)	63 (24 - 16 - 23)	89 (31 - 16 - 42)
14	157 (30 - 47 - 80)	40 (21 - 10 - 9)	65 (25 - 16 - 24)	90 (31 - 16 - 43)
13	161 (30 - 50 - 81)	43 (23 - 10 - 10)	67 (26 - 16 - 25)	93 (32 - 18 - 43)
12	170 (31 - 58 - 81)	43 (23 - 10 - 10)	68 (26 - 17 - 25)	96 (32 - 20 - 44)
11	178 (31 - 64 - 83)	43 (23 - 10 - 10)	69 (26 - 18 - 25)	98 (32 - 20 - 46)
10	187 (32 - 72 - 83)	46 (24 - 10 - 12)	75 (28 - 20 - 27)	105 (33 - 25 - 47)
9	197 (32 - 79 - 86)	49 (24 - 12 - 13)	77 (28 - 21 - 28)	110 (33 - 29 - 48)
8	211 (32 - 93 - 86)	52 (25 - 13 - 14)	78 (29 - 21 - 28)	114 (34 - 31 - 49)
7	224 (33 - 104 - 87)	56 (27 - 14 - 15)	82 (30 - 22 - 30)	120 (36 - 33 - 51)
6	242 (35 - 120 - 87)	58 (27 - 16 - 15)	83 (30 - 23 - 30)	124 (36 - 34 - 54)
5	258 (36 - 134 - 88)	59 (28 - 16 - 15)	84 (30 - 23 - 31)	125 (37 - 34 - 54)
4	273 (36 - 148 - 89)	60 (28 - 17 - 15)	87 (30 - 25 - 32)	129 (37 - 37 - 55)
3	304 (36 - 179 - 89)	63 (28 - 18 - 17)	89 (30 - 26 - 33)	129 (37 - 37 - 55)
2	362 (37 - 236 - 89)	70 (30 - 22 - 18)	94 (32 - 29 - 33)	133 (38 - 40 - 55)
1	454 (41 - 321 - 92)	78 (30 - 30 - 18)	103 (33 - 37 - 33)	151 (42 - 52 - 57)
0	507 (45 - 369 - 93)	507 (45 - 369 - 93)	507 (45 - 369 - 93)	507 (45 - 369 - 93)

4.3.1. Scenario I - Removing only *AGC*

In this scenario, all `NCF_sdhash` versions had better results than `sdhash` for $t \leq 21$ considering precision (see Fig. 1). The best setting was $N > 20$ with many undesired matches being removed and many templates considered as tp due to the small number of files sharing the same layout. `sdhash` had the worst results, where the decrease of t had a negative impact due to a large number of fp matches. On the other hand, removing the common features resulted in many undesirable matches being ignored, having a less significant impact on `NCF_sdhash` (except for $N > 100$) when decreasing t.

Considering now the Recall metric (Fig. 2), no algorithm found all similar matches. `sdhash` and `NCF_sdhash` with $N > 20, 50, 100$ had the best results. For this metric, we had a bad influence of templates for `NCF_sdhash` with low N settings. Many template matches were removed from results due to the limited number of models in the database. As N increased, the features related to templates were not considered common anymore, and the matches became relevant again. For $N > 3$, we found only 33/93 template matches, while for $N > 20$ we had 91/93 ($t = 1$).

Table 8. Number of file matches per score and per class - Part II: NCF_sdhash ($N > 10, 20, 50, 100$).

Score >=	File Matches in the form: *#matches (UGC - AGC - TC)*			
	N >10	**N >20**	**N >50**	**N >100**
21	98 (26 - 13 - 59)	108 (26 - 11 - 71)	111 (26 - 13 - 72)	112 (26 - 15 - 71)
20	102 (27 - 14 - 61)	113 (26 - 14 - 73)	114 (26 - 15 - 73)	114 (26 - 16 - 72)
19	105 (28 - 16 - 61)	115 (27 - 15 - 73)	115 (27 - 15 - 73)	116 (27 - 16 - 73)
18	106 (29 - 16 - 61)	117 (28 - 16 - 73)	119 (28 - 17 - 74)	118 (28 - 17 - 73)
17	110 (30 - 16 - 64)	122 (29 - 17 - 76)	123 (29 - 18 - 76)	122 (29 - 17 - 76)
16	114 (31 - 18 - 65)	125 (31 - 18 - 76)	124 (30 - 18 - 76)	123 (30 - 17 - 76)
15	115 (31 - 19 - 65)	127 (31 - 19 - 77)	125 (30 - 19 - 76)	125 (30 - 19 - 76)
14	116 (31 - 19 - 66)	128 (31 - 19 - 78)	127 (30 - 19 - 78)	128 (30 - 20 - 78)
13	118 (33 - 19 - 66)	129 (32 - 19 - 78)	128 (30 - 19 - 79)	133 (31 - 23 - 79)
12	122 (33 - 22 - 67)	134 (33 - 22 - 79)	132 (30 - 22 - 80)	139 (31 - 28 - 80)
11	124 (33 - 23 - 68)	137 (33 - 23 - 81)	137 (31 - 25 - 81)	142 (31 - 30 - 81)
10	130 (33 - 27 - 70)	143 (33 - 28 - 82)	143 (31 - 29 - 83)	148 (31 - 35 - 82)
9	135 (34 - 30 - 71)	147 (34 - 30 - 83)	147 (32 - 31 - 84)	155 (32 - 40 - 83)
8	138 (34 - 33 - 71)	151 (34 - 32 - 85)	151 (32 - 33 - 86)	162 (33 - 44 - 85)
7	142 (36 - 34 - 72)	158 (36 - 35 - 87)	161 (35 - 37 - 89)	173 (35 - 50 - 88)
6	144 (36 - 35 - 73)	159 (36 - 35 - 88)	165 (36 - 40 - 89)	181 (36 - 57 - 88)
5	146 (37 - 36 - 73)	163 (37 - 38 - 88)	172 (37 - 46 - 89)	188 (36 - 64 - 88)
4	148 (37 - 38 - 73)	166 (37 - 40 - 89)	176 (37 - 49 - 90)	212 (37 - 86 - 89)
3	151 (37 - 41 - 73)	169 (38 - 42 - 89)	190 (38 - 62 - 90)	231 (38 - 104 - 89)
2	158 (38 - 46 - 74)	178 (39 - 50 - 89)	203 (39 - 74 - 90)	247 (39 - 119 - 89)
1	171 (41 - 55 - 75)	188 (42 - 55 - 91)	222 (42 - 89 - 91)	265 (42 - 133 - 90)
0	507 (45 - 369 - 93)	507 (45 - 369 - 93)	507 (45 - 369 - 93)	507 (45 - 369 - 93)

Fig. 3 shows the results for the F_1 score. We had the best combination between precision and recall for $N > 20$. We could see that `sdhash` results degraded significantly for threshold $t < 21$ due to the high number of false positives (bad precision). On the other hand, low t values increased the performance of `NCF_sdhash` (with low and mid N values). However, mid N values are the ones recommended when template matches are relevant for investigations. Besides, using $t > 0$ showed to be beneficial and should be taken into consideration when working with `NCF_sdhash`.

4.3.2. Scenario II - Searching for *UGC* only

For the second scenario, we are interested in finding only *UGC* matches. Fig. 4 shows our results regarding precision. Notice that low N values stood out in this experiment since they were responsible for removing many template matches - the lower the N, the better the precision. `sdhash` had the worst results since it detected many *TC* matches as relevant, once it does not distinguish the class of similarity. All algorithms had low precision values mostly due to templates, which were harder to remove even for `NCF_sdhash` (see tables 7 and 8). In most cases, decreasing to $t = 1$ had a small impact on precision.

The recall rates are shown in Fig. 5. The worst results were obtained for low N values, in which we could not identify a few similar files with too many changes (differences). Besides, we found a particular case where six comparisons of `html` files were

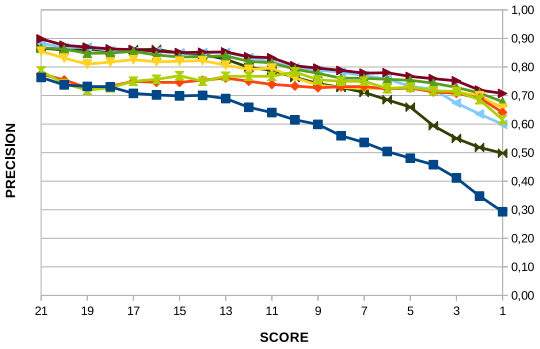

Figure 1. Scenario I: Precision vs. score results

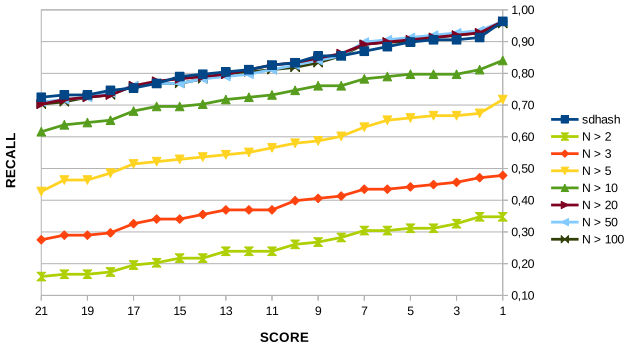

Figure 2. Scenario I: Recall vs. score results

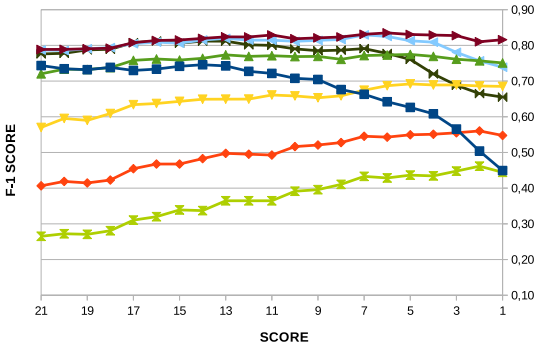

Figure 3. Scenario I: F_1 vs. score results

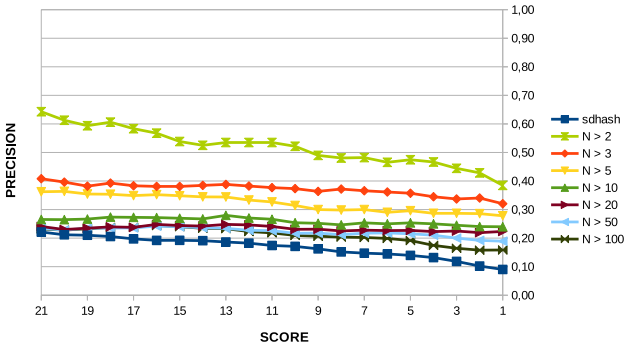

Figure 4. Scenario II: Precision vs. score results

between identical objects in our database. This way, all features related to them became common for some NCF_sdhash settings; $N > 2$ and $N > 3$ produced $s = 0$, while sdhash and others had $s = 100$. As t decreased, we had similar results for sdhash and other NCF_sdhash settings. For recall, it is worth to accept small scores since many additional matches were found; most cases reached more than 90% at $t = 1$, while for the recommended $t = 21$, they had about 60%.

Given the F_1 score results (Fig. 6), we can conclude that for scenarios where templates matches are not relevant, low N values are recommended for its good balance between precision and recall. sdhash had the worst results which degraded significantly for $t < 21$. For NCF_sdhash with low/mid N settings, it is worth looking for matches with $t \geq 1$ since even low values tend to present relevant matches.

4.3.3. Scenario III - No template matches

The third scenario does not consider template matches and seeks to analyze how they influenced precision and recall rates. NCF_sdhash was superior regarding precision (Fig. 7). Normally, the lower the N, the better the precision. Low N values in NCF_sdhash are more prone to remove *AGC* (see tables 7 and 8) since many features repeating in a few files are considered common. An example is a match of two pdf files where sdhash and some NCF_sdhash settings detected as similar, but the files were

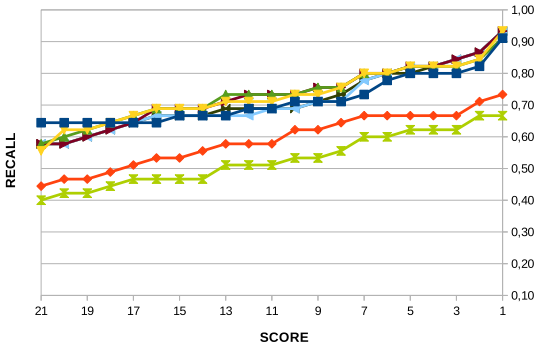

Figure 5. Scenario II: Recall vs. score results

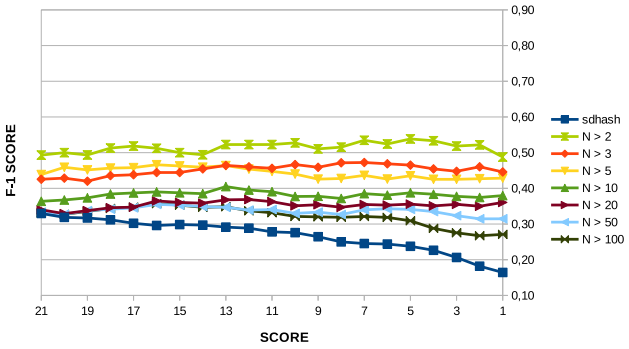

Figure 6. Scenario II: F_1 vs. score results

different. By using $N > 2$ or $N > 3$, we could remove many features shared by those files and with a few other ones (e.g., a feature related to a font specification) and have $s = 0$. As we decrease t, `sdhash` results dropped from 50% to 10%, a significant degradation on its performance. Lower values of N presented a less aggressive degradation on these precision results due to the small number of false positives compared to `sdhash`.

For recall (Fig. 8), we had the same results as scenario II since in both cases we ignored *TC* matches. Finally, the results of F_1 score (Fig. 9) for decreasing t showed a poor performance of `sdhash` again, while all settings of `NCF_sdhash` with low/mid N values had similar/better results. For these settings, no significant degradation was noticed for low t values, and considering them as relevant results can be beneficial.

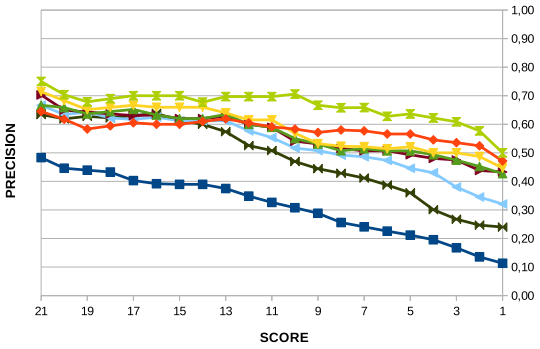

Figure 7. Scenario III: Precision vs. score results

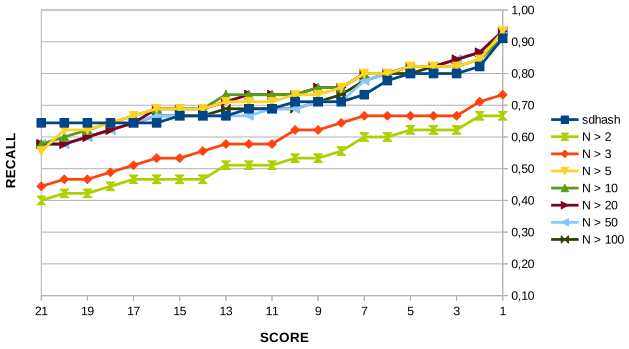

Figure 8. Scenario III: Recall vs. score results

5. Discussion

Based on the experiments described in the previous section, we here discuss the lessons learned, specifically, the correlation of removing common features and the similarity score with respect to the different scenarios.

RQ1. How does the removal of common features impact digital forensics investigations for the different classes of similarity?

By removing the common features, many *AGC* matches were filtered out. *TC* matches were a problem for low N-values in `NCF_sdhash`, as well as finding a few *UGC* matches

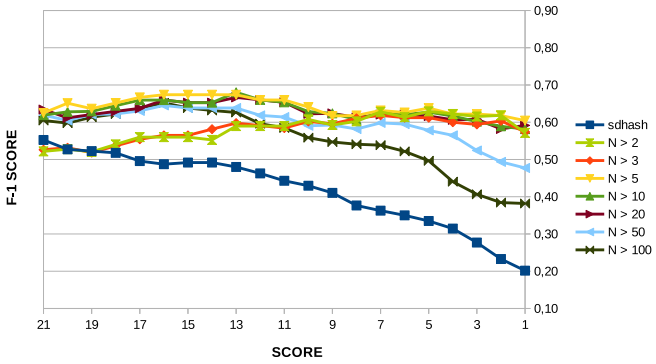

Figure 9. Scenario III: F_1 vs. score results

of high degree of complexity (too many modifications on the files content). Mid/high-values of N in `NCF_sdhash` had similar or better results than `sdhash` in all cases.

With respect to template matches, we had many instances of a few files sharing the same layout in our database. Low N-values in `NCF_sdhash` removed many of these matches from the results but also kept some of them. Consequently, scenario I and II were impacted negatively, where we could neither identify nor remove all *TC* matches effectively. We can confirm this assumption by observing the increase in recall (scenario I vs. scenario III) and precision (scenario II vs. scenario III). In the first case, recall dropped significantly since many templates were removed, while in the second case, some *TC* matches that we could not remove from the results were detected as fp.

`NCF_sdhash` with low N settings (except for $N > 5$) also underperformed in the detection of a few matches between files with too many modifications (differences). Besides, some identical files found in the database contributed for degradation in results (see Sec. 4.3.2). `sdhash` and all settings with $N > 5$ had more than 91.00% of recall. `NCF_sdhash` using $N > 2$ and $N > 3$ had 66.67% and 73.33%, respectively. The reason for higher recall rates of `sdhash` and most versions of `NCF_sdhash` is due to the common features. By removing them, some instances had $s = 0$ since the number of features related to *UGC* were too small or nonexistent. In such cases, the match may happened because common features were still present in the similarity assessment.

In short, `sdhash` tended to detect many matches related to template/application-generated content and `NCF_sdhash` with low N-values is inadequate for template detection or comparisons with a high level of complexity, which do not rely on the common features. Although mid/high values of N could perform well for template detection in this particular data set, future work is necessary to separate this sort of match since it is hard to know the number of files sharing the same template in a large set.

RQ2. How are precision/recall rates affected by the removal of common features?

In general, removing the common features increased precision, and for low N values, decreased recall. `sdhash` performed the worst regarding precision in all scenarios due to the presence of the common features. `NCF_sdhash` had the best results for low N values, except for scenario I, where mid-values of N had the best results for detecting many *TC* matches. For recall, mid/high-values of N had similar/better results than `sdhash`. Low

N-values for `NCF_sdhash` performed worse, especially for scenario I where many *TC* matches were missed. Given the F_1 score (balance between precision and recall), we can see that `sdhash` performance dropped significantly in all scenarios with the decrease of t. On the other hand, low/mid N values of `NCF_sdhash` presented the best results as t dropped to 1.

In our experiments, using $N > 20$ had better results than higher N values ($N > 50$ or $N > 100$) for all cases. We believe that no further benefit is achieved for higher values of N. The number of tp was about the same with a significant reduction in the number of fp. Besides, when considering template similarity as a relevant result (scenario I), $N > 20$ should also be the one adopted given its better precision/recall rates.

RQ3. How is the recommended threshold value of 21 for `sdhash` affected by removing common features?

Our experiments revealed that many matches were left out by using $t = 21$ for `sdhash`. From the 45 *UGC* matches, only 29 were found. Using $t = 1$ allowed us to increase the number of *UGC* to 41 at the cost of many additional *AGC* matches (321/369). *TC* were also benefited from using lower t values (92/93). Although recall increased, precision dropped significantly for the threshold reduction. Besides, F_1 score showed that it is not worth using $t < 21$ for `sdhash` due to the significant degradation in its performance.

On the other hand, `NCF_sdhash` showed improvements when choosing lower threshold values. The best performances of all settings were with $N > 20$, where many *UGC* and *TC* were found (42/45 and 91/93, respectively) at a small cost of *AGC* matches (55/369) compared to `sdhash`. F_1 score values showed minor improvements when using low t values. For this reason, we believe that low thresholds should be considered for digital forensic investigations given that many relevant matches were found. For $N > 2$, the best F_1 value was using $t = 2$ (scenario I) and $t = 5$ (scenarios II and III), while for $N > 20$ we had $t = 6$ (scenario I), $t = 12$ (scenario II), and $t = 13$ (scenario III).

Remark: The accuracy, defined as $(tp+tn)/(tp+tn+fp+fn)$, is not presented here, since `sdhash` and all `NCF_sdhash` versions had similar values ($> 99.00\%$) due to the enormous number of true negatives pointed out by both algorithms.

6. Conclusions & Future work

In this paper, we analyzed the impact of excluding the common features from digests created with approximate matching (AM) functions. Our results showed that many matches only happened due to the common features, and by their removal, we achieved a significant reduction in the number of matches. This practice can also benefit precision/recall rates, where different settings will aid each metric differently based on the goal of the investigation. We also analyzed the impact on the threshold score of AM and verified that all scores produced relevant matches for a small cost in the number of false positives. Future work consists of analyzing other data sets to confirm if the N-values used in this work can be adopted universally. Also, we would like to study other possible ways of identifying template content similarity.

7. Acknowledgment

This work is partially supported by CAPES FORTE Project (23038.007604/2014-69).

References

[Bloom 1970] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

[Breitinger and Baier 2013] Breitinger, F. and Baier, H. (2013). *Similarity Preserving Hashing: Eligible Properties and a New Algorithm MRSH-v2*, pages 167–182. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Breitinger et al. 2014] Breitinger, F., Guttman, B., McCarrin, M., Roussev, V., and White, D. (2014). Approximate matching: definition and terminology. *NIST Special Publication*, 800:168.

[Foster 2012] Foster, K. (2012). Using distinct sectors in media sampling and full media analysis to detect presence of documents from a corpus. Technical report, Naval Postgraduate School Monterey (CA).

[Garfinkel and McCarrin 2015] Garfinkel, S. L. and McCarrin, M. (2015). Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. *Digital Investigation*, 14:S95–S105.

[Gutierrez-Villarreal 2015] Gutierrez-Villarreal, F. J. (2015). Improving sector hash carving with rule-based and entropy-based non-probative block filters. Technical report, Naval Postgraduate School Monterey (CA).

[Kornblum 2006] Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, 3:91–97.

[Moia et al. 2019] Moia, V. H. G., Breitinger, F., and Henriques, M. A. A. (2019). The impact of excluding common blocks for approximate matching. pages 1–11. TO BE PUBLISHED.

[Oliver et al. 2013] Oliver, J., Cheng, C., and Chen, Y. (2013). TLSH–a locality sensitive hash. In *Cybercrime and Trustworthy Computing Workshop (CTC), 2013 Fourth*, pages 7–13. IEEE.

[Olson and Delen 2008] Olson, D. L. and Delen, D. (2008). *Advanced data mining techniques*. Springer Science & Business Media.

[Raff and Nicholas 2018] Raff, E. and Nicholas, C. (2018). Lempel-ziv jaccard distance, an effective alternative to ssdeep and sdhash. *Digital Investigation*, 24:34–49.

[Roussev 2010] Roussev, V. (2010). Data fingerprinting with similarity digests. In *IFIP International Conf. on Digital Forensics*, pages 207–226. Springer.

[Roussev 2011] Roussev, V. (2011). An evaluation of forensic similarity hashes. *Digital investigation*, 8:34–41.