Computing and Informatics, Vol. 36, 2017, 887–907, doi: 10.4149/cai_2017_4_887

EVALUATION AND IMPLEMENTATION OF *N*-GRAM-BASED ALGORITHM FOR FAST TEXT COMPARISON

Maciej Wielgosz, Paweł Szczepka, Paweł Russek
Ernest Jamro, Kazimierz Wiatr

AGH University of Science and Technology
Mickiewicza 30 Av., 30-059 Krakow, Poland
e-mail: {wielgosz, russek, jamro, wiatr}@agh.edu.pl

Marcin Pietroń, Dominik Żurek

ACC Cyfronet AGH
Nawojki 11, 30-950 Krakow, Poland
e-mail: {marcin.pietron, dominik.zurek}@cyfronet.pl

Abstract. This paper presents a study of an n-gram-based document comparison method. The method is intended to build a large-scale plagiarism detection system. The work focuses not only on an efficiency of the text similarity extraction but also on the execution performance of the implemented algorithms. We took notice of detection performance, storage requirements and execution time of the proposed approach. The obtained results show the trade-offs between detection quality and computational requirements. The GPGPU and multi-CPU platforms were considered to implement the algorithms and to achieve good execution speed. The method consists of two main algorithms: a document's feature extraction and fast text comparison. The winnowing algorithm is used to generate a compressed representation of the analyzed documents. The authors designed and implemented a dedicated test framework for the algorithm. That allowed for the tuning, evaluation, and optimization of the parameters. Well-known metrics (e.g. *precision*, *recall*) were used to evaluate detection performance. The authors conducted the tests to determine the performance of the winnowing algorithm for obfuscated and unobfuscated texts for a different window and n-gram size. Also, a simplified version of the text comparison algorithm was proposed and evaluated to reduce the computational

complexity of the text comparison process. The paper also presents GPGPU and multi-CPU implementations of the algorithms for different data structures. The implementation speed was tested for different algorithms' parameters and the size of data. The scalability of the algorithm on multi-CPU platforms was verified. The authors of the paper provide the repository of software tools and programs used to perform the conducted experiments.

Keywords: Text similarity analysis, n-gram-based model, GPGPU implementation, multi-CPU implementation

1 INTRODUCTION

Nowadays, a vast amount of data is generated by millions of electronic media sources. It is estimated that all the data collected on the internet domain amounted to 2.7 ZB in 2012. It is a 48 % growth if compared to 2011. At the end of 2013, this number reached 4 ZB [1, 2]. As a result, an internet user can quickly find information in any area and subject he is interested. Today, researchers and students need reliable knowledge and information sources which are accessible as soon as possible. Simultaneously, data is an essential source of knowledge for companies, businesses, and individual people. Unfortunately, the free and unlimited access to digitally stored information causes new problems. One of them is the widespread usage of dishonest web material and author's rights violation. A class of the above phenomena is plagiarism, which affects science and education mainly. Plagiarism raises the need for developing cheap and robust plagiarism detection tools and services. Unfortunately, the building and running of an anti-plagiarism system requires substantial investments and funds.

Web search operations for the most frequent Polish word: 'się', returns over 600 million web pages. Consequently, the number of pages in the Polish language that are indexed in the web and should be analyzed by the anti-plagiarism system is significant. The average number of words for a web page is over 2 400 words, i.e. approximately two pages. The number of Polish students who graduate is nearly 0.5 million each year. Let us assume that every student writes 50 pages of his final thesis. Thus, roughly 25 million pages are generated each year. Summing up, a national center for plagiarism detection should be able to store at least 600 million pages, and 25 million pages should be added every year. Rough approximation presented later in this article shows that the straightforward comparison of eight pages with the database containing 0.6 billion documents requires roughly 1 280 seconds. Let us assume that servers power consumption is 200 W, and an energy price is \$ 0.2 per one kWh. Consequently, one has to spend \$ 45 000 to compare 25 million pages against a 600 million document database ($\$\,45\,000 \approx 25\,000\,000 * (1\,280/3\,600)\,\mathrm{h} * 0.2\,\mathrm{kW} * \$\,0.2\,\mathrm{kWh}^{-1}/8$ pages). As we can see, the cost of running the system can be substantial and adequate algorithms should be chosen very carefully.

The primary focus of the authors' study is to examine the n-gram-based algorithm and its implementation, which would be useful to build a system capable of comparing millions of files in a short time. The system should be cheap in terms of computational and storage requirements. Therefore, the authors examined the feasibility of using the winnowing method [3, 4] and its different implementations to address this goal.

2 TEXT SIMILARITY SEARCH AND FILE COMPARISON

Plagiarism detection is a particular kind of document analysis, which is focused on similarity detection. It uses the very same set of tools as other techniques in the field. Research on duplication detection has been examined in several papers, e.g. [5, 6, 7]. There are two major approaches to document similarity detection: external and intrinsic. The first one covers all the material resources, which are available in a given database. The overall detection procedure has been devised and well described in [7]. It is built around the three steps: the heuristic retrieval step, a detailed analysis step, and the post-processing step. The intrinsic approach focuses only on the text at hand. That means that no external documents are necessary to be used. The text is processed with a high focus on inconsistencies, which may indicate that using external sources have been used to compile the document. This task is difficult, highly sophisticated, and it requires that the system can recognize writing style changes in the text. Consequently, the intrinsic approach is not in practical use due to its imperfections.

Plagiarism detection systems are presented in [7]. They are built on architectural and algorithmic foundations which are reflected in data representation. Namely, two models are dominant: the Vector Space Model (VSM) [8] and the Boolean model [9] but there are also some other approaches such as Document Occurrence Representation [10]. Table 1 in [7] presents the performance comparison of the systems, and there are two algorithms based on n-grams among six enumerated in this table. These systems were firstly classified in [11] and achieved an *f-measure* of 0.69 and 0.61, respectively. The VSM that is based on n-grams proved to yield the best results. There has been a huge leap forward in the course of recent years, which is reflected in a rising similarity detection efficiency of the systems taking part in the PAN competition. The results of consecutive years can be found in [7, 12, 13, 14, 15].

In the algorithmic study part of this paper examines winnowing, the n-gram-based method, and its effectiveness in the mapping from the text to the document comparison space. This process of mapping directly affects comparison results. In other words, the authors show the relationships between the choice of a fingerprint and detection quality. Detection of exact matches is a relatively easy task, but it is not satisfactory in some cases. For instance, it would miss similarities where documents differ in minor details resulting from intentional modifications or artifacts introduced by text-to-feature transformation. Moreover, it would miss most of the important textual relationships between documents. The above problem can be

mitigated if the algorithms parameters are properly tuned. The most important part of the plagiarism detection method is a choice of the right criterion for notification of a plagiarism occurrence in the examined record. The particular threshold of a similarity value must be set to trigger a plagiarism alert. Also, it must be decided if the similarity value is measured with respect to the aggregated all-documents data set or in the one-to-one document manner. Since the comparison scenario affects the system performance, it will also be an important part of the paper.

3 WINNOWING ALGORITHMS

An n-gram is a continuous substring of n letters, which is selected from an original document. The number of n-grams that are generated from a document is large, and it is roughly equal to the number of letters in a paper. It can be calculated precisely from the Equation (1).

$$N = (n - k + 1) \tag{1}$$

where N – total number of n-grams, n – number of letters in a document, k – size of an n-gram.

Using all the document's n-grams for a file comparison would result in an overwhelming number of computations. Therefore, only a fraction of n-grams, which are denoted as fingerprints are considered. The choice of a proper n-gram's subset, raises the question of adequate selection policy, i.e. which n-grams should be selected to represent the document content. The patterns of the selection algorithm impact significantly the later result of files comparison. Several approaches from work [16] were adopted in [17]. These are *the 0 mod p*, *the minimum*, and *the maximum* value in a given range. Respectively, the authors proposed choosing the smallest n-gram's hashes as fingerprints in [3].

Fixing the number of n-grams per document makes the system more scalable, but does not guarantee that materials significantly different in size can be compared meaningfully. It is important to provide a balanced text coverage, i.e. the fingerprint must be composed of n-grams (hashes) uniformly distributed over the whole document. It makes the algorithm robust to obfuscation and straightforward rephrasing. Therefore, in this paper the winnowing procedure was employed for fingerprints selection.

The idea behind the algorithm is to represent a document's text in a fingerprint uniformly. The detailed description of winnowing algorithm is included in [3]. Here only a brief outline will be presented. Winnowing starts with hash the generation for all the text's n-grams. Consequently, it creates a subset of text's hashes $\{h(1), h(2), h(3), \ldots, h(w)\}$ where w is called a window size. The sliding window techniques is used to produce all the subsets. Winnowing guarantees the detection of similar text fragments, which are at least $w + k - 1$ letters long. To maintain the above guarantee, it is necessary that at least one hash from every window of the size w contributes to the fingerprint.

The following example illustrates the idea behind the algorithm. Given a sample set of hashes: [26 122 19 46 88 42 19 47 111 64 28 64 65 28 38 11 17 110 112] and window size $w = 5$ the following steps are to be conducted in order to form a fingerprint: (26 122 **19** 46 88), (122 19 46 88 42), (19 46 88 42 **19**) (46 88 42 19 47), (88 42 19 47 111), (42 19 47 111 64), (19 47 111 64 28), (47 111 64 **28** 64), (111 64 28 64 65) (64 28 64 65 **28**), (28 64 65 28 38), (64 65 28 38 **11**), (65 28 38 11 17), (28 38 11 17 110), (38 11 17 110 112). Consequently, the fingerprint is composed of a set of hashes: [**19 19 28 28 11**].

Figure 1 shows the scheme of the document's fingerprint generation. The input text is pre-processed to eliminate unimportant information in the first step. N-grams are selected, and their hashes computed in the subsequent steps. The final stage is the most important one since it generates the document's fingerprint. The files' similarity criteria are based on inclusion measures given by the Formula (2) and Formula (3).

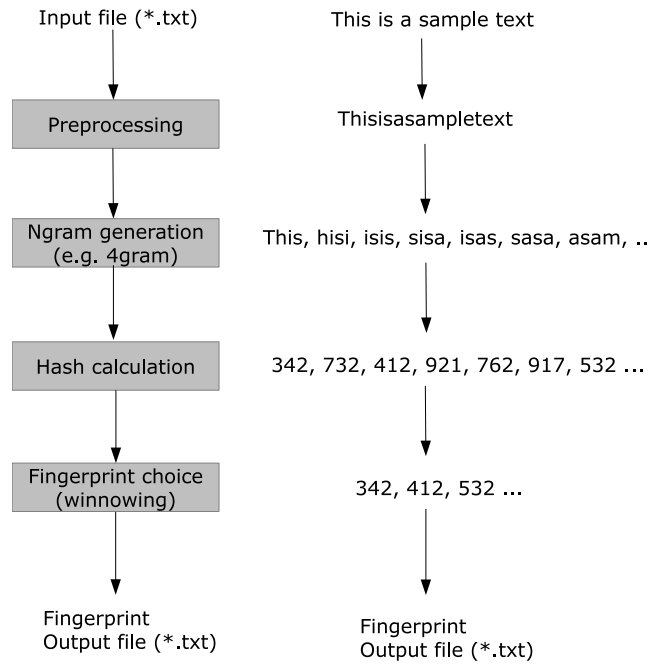


Figure 1. The *N-gram* generation scheme (n-gram size: 4, window size: 4)

$$\text{similarity}(A, B) = |Ha \cap Hb|/|Ha| \tag{2}$$

$$\text{similarity}(B, A) = |Ha \cap Hb|/|Hb| \tag{3}$$

where H_a and H_b are cardinalities of A and B documents' hash sets.

Consequently, the following steps should be accomplished to compare two documents:

- selection of the n-gram size,
- computation of the hashes of all the n-grams,
- fingerprint generation from the hashes' set,
- documents similarity computation using the respective measures (Equation (2) or (3)).

4 EXPERIMENTS AND DISCUSSION

The authors carried out several experiments to evaluate the winnowing method performance in terms of both detection quality and computation workload. The detection quality was determined using a corpus provided by Bauhaus-Universität Weimar for the PAN competition [18]. The proposed test framework assumes that the architecture of the duplication detection algorithm is composed of two sections. Those are the retrieval and alignment sections, as it was described in [7, 12, 13, 14, 15].

The three well-known metrics are used to evaluate the detection quality of the algorithm in the PAN competition [7]:

$$\text{precision} = \frac{(\text{relevant pairs}) \cap (\text{retrieved pairs})}{(\text{retrieved pairs})}, \tag{4}$$

$$\text{recall} = \frac{(\text{relevant pairs}) \cap (\text{retrieved pairs})}{(\text{relevant pairs})}, \tag{5}$$

$$\text{f-measure} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{6}$$

The role of the retrieval stage is to select the subset of documents that will be analyzed in detail in the alignment phase. The algorithms employed in the first stage are not complex so they can handle all the input files in a reasonable time. The system throughput and *recall* for the retrieval step should be high. Great *recall* rate ensures that a small number of documents are missed for the processing in the alignment stage. The role of the alignment stage is a detailed analysis of the documents selected in the retrieval stage and denoted as suspicious. It is worth noting that low *recall* at the retrieval stage results in a substantial document loss that is unrecoverable. However, the alignment phase can make up for poor *precision* at the expense of higher computational effort.

The authors designed and implemented the test environment that is based on the scripts delivered by PAN. It was used together with the corpus built from the files that belong to four different categories:

- no obfuscation,
- random obfuscation,
- translation obfuscation,
- summary obfuscation.

PAN's 2013 test database contains 1 827 suspicious and 3 230 source documents, which are composed of:

- 1 000 cases of no obfuscation plagiarism,
- 1 000 cases of random obfuscation,
- 1 000 cases of transformation plagiarism,
- 1 185 cases of summary obfuscation.

The test script generates a report (a file with tags) for each document. The report links a suspicious file with one or several original records. The tags are used to compute the comparison quality measures that are given by Equations (4), (5), and (6). For the winnowing algorithm, the authors had adopted several test scenarios for both obfuscated and unobfuscated documents.

4.1 Winnowing with No Obfuscation

The results of detection quality tests that were conducted are given in Figures 2 and 3. The goal of the experiments was to find the threshold value, n-gram width, and window size, which yield the best results in terms of the three performance metrics. Here, the threshold is defined as the fraction of hashes, which are repeated in the compared files that classifies them as duplicates. The figures show that the experiment results match the theoretical expectations, i.e. *recall* drops with an increase in the threshold. Precision is inversely proportional to the *recall*. The results show that the more similarity is expected between documents (i.e. the threshold is set higher), the fewer files are falsely categorized as duplicates, but fewer duplicates are missed (larger *precision*). In the case of the two-stage system, which is composed of the retrieval and alignment phase such a phenomenon means that for the higher threshold, fewer files pass the alignment phase. On the other hand, the lower similarity (the lower the threshold) of two documents is the required in retrieval step, the more materials are thrown into the alignment phase.

Interestingly, *precision* is only slightly affected by the window size, which means that there is very little performance penalty (loss of detection quality loss) for a big window size parameter. Consequently, there is a relatively modest impact on the window size of *f-measure*, which is mainly determined by the threshold. Consequently, the use of large windows does not degrade the *f-measure* (recognition efficiency) for unobfuscated documents. Thus, it is a useful conclusion, to use large windows to reduce the fingerprint size and overall system throughput performance.

Figures 4, 6, and 5 present how the gram size, window size, and threshold affect *recall*, *precision*, and *f-measure* for unobfuscated winnowing. The figures clearly

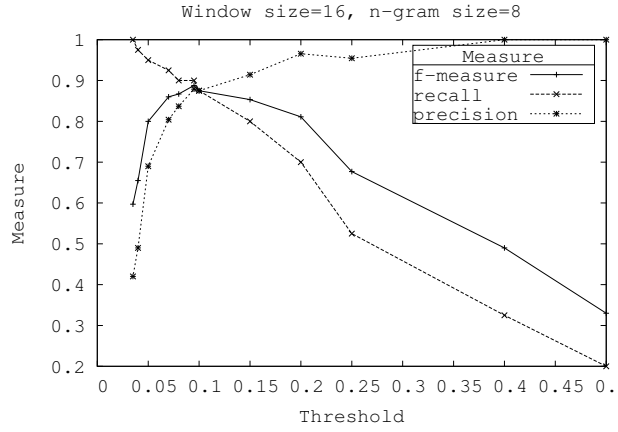


Figure 2. Winnowing with no obfuscation, *n-gram* size = 8, Window size = 16

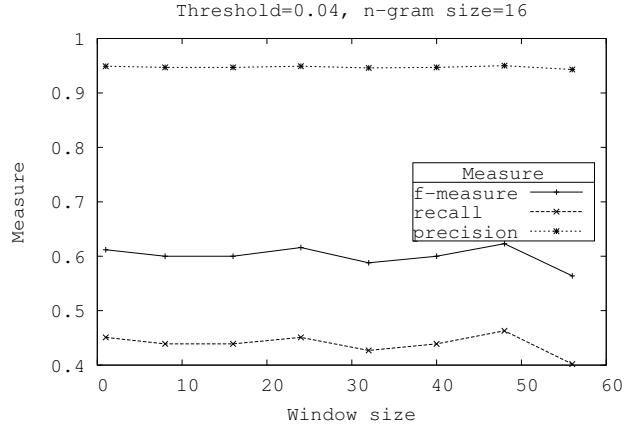


Figure 3. Winnowing with no obfuscation, *n-gram* size = 32, Threshold = 0.04

depict that the threshold has a much more significant impact on the performance of the algorithm than the n-gram size. According to the conducted experiments, the best results were obtained for window and n-gram size of 40 and 32, respectively.

4.2 No-Threshold Experiment

This experiment was conducted to verify the hypothesis that *recall* will converge to one if a large n-gram size is chosen and the threshold is set to 0. For this analysis, a long n-gram instead of a set of shorter n-grams is used, and a long enough piece of text should overlap in the examined documents to detect duplication. This experi-

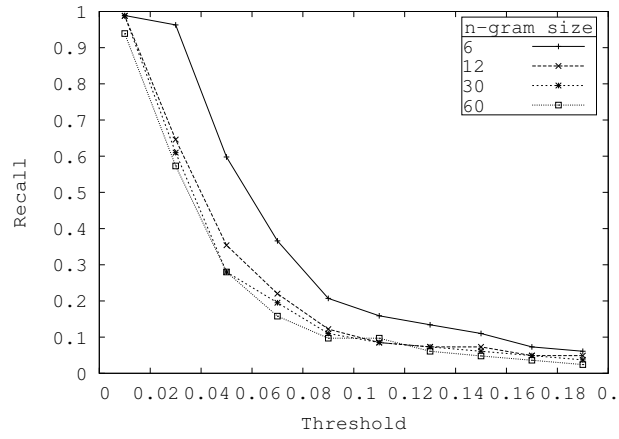


Figure 4. *Recall* for the unobfuscated winnowing (window size is 40)

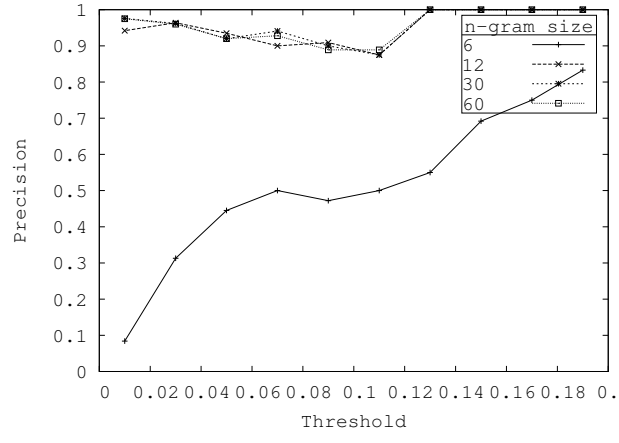


Figure 5. *Precision* for the unobfuscated winnowing (window size is 40)

ment was necessary to verify the accuracy of the fastest implementation presented in Section 5.3. The text sample in documents is considered a duplicate if the n-grams of a large size match. The threshold eliminates the need for additional computation of overlapped text percentage and substantially increase the performance of the proposed implementation. Just one text overlap occurrence will trigger the similarity flag.

Table 1 shows that the results in terms of *recall* are better than in the previous cases. Recall roughly reaches one, which means that all the duplication pairs were detected. This approach is superior to the previous one because its efficiency does

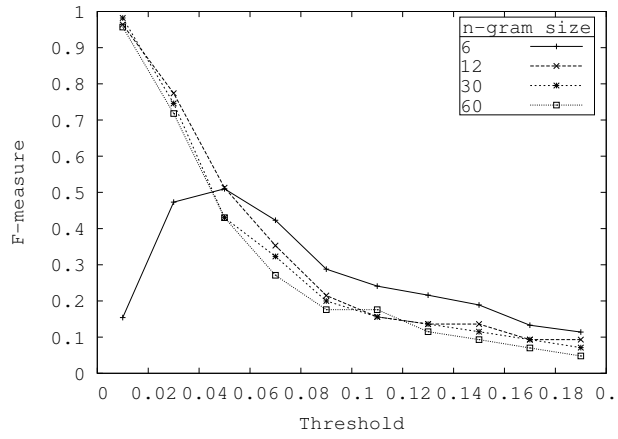


Figure 6. *F-measure* for the unobfuscated winnowing (window size is 40)

(Ngram; Window)	threshold	precision	recall	f-measure
(200; 40)	0	0.691	0.912	0.786
(100; 40)	0	0.691	1	0.817

Table 1. Winnowing, no obfuscation

not depend on the document length. In the standard implementation, a very low threshold should be set to obtain proper results. That rule means that for short documents fewer n-grams are sufficient to qualify them as similar. The document length issue can be solved by changing the similarity measures (Equations (2) and (3)).

4.3 Winnowing with Obfuscation

The previous sections present the results of unobfuscated duplication detection, which involves simple text copying without any modifications. Such type of plagiarism is very common, however in some cases the text is modified, e.g. the order of words or sentences is changed. Sometimes different phrasing and tenses are to preserve the original meaning while changing the actual words and sentences.

Figure 7 presents the results obtained for the obfuscated documents comparison. *Recall* is higher for smaller window size values (6 or 8). The results for obfuscated material depend on this algorithm's parameter because an obfuscation breaks long similarity patterns which exist in duplication with no obfuscation. However, it is possible to find the duplication pairs, which do not correlate with high *precision*. For *recall* that is close to one, *precision* is approximately 0.5 which requires an efficient alignment phase to process twice as much data as in an ideal retrieval case. Furthermore, this number is expected to drop with a rise the number of documents in the test corpus.

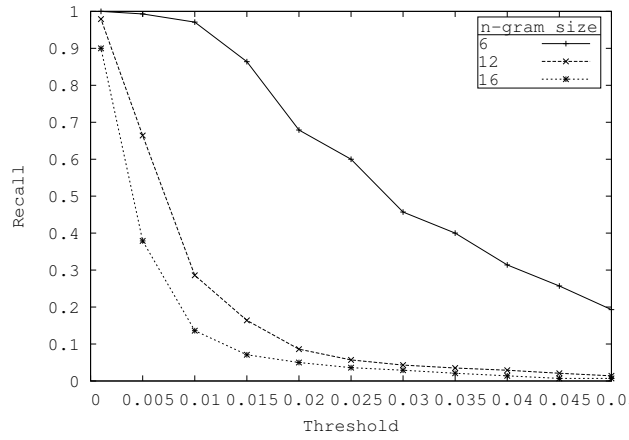


Figure 7. *Recall* as a function of the n-gram size and the threshold for the obfuscated winnowing

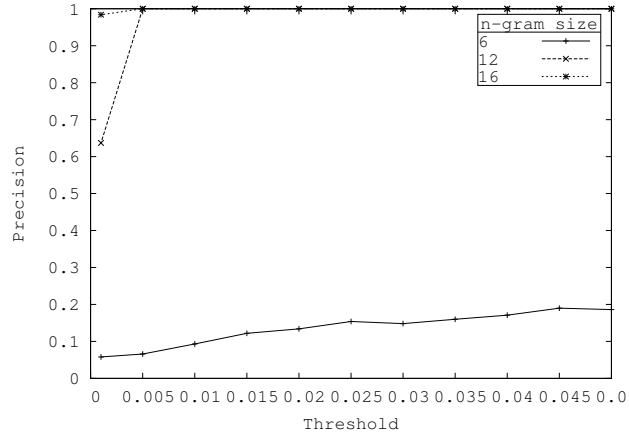


Figure 8. *F-measure* as a function of an n-gram size and threshold for the obfuscated winnowing

5 IMPLEMENTATION RESULTS

Two of the most critical sections of the winnowing algorithm were extracted to speed up the computation and limit system power consumption simultaneously. The authors optimized the implementation of fingerprint generation and fingerprint comparison that are the algorithm's computational kernels. The implementations were prepared for Intel's many-CPU (Xeon E5645, 2.40 GHz) and GPGPU (Nvidia's Tesla M2090) processors compiled with -O3 flags. This section provides results of

both approaches with a particular focus on the selection of the right platforms for each task. A series of tests were conducted for different architectures, and the same structure of benchmark data was used. Each document in the set was composed of 8 192 hashes, and each hash was four bytes long.

The number of hashes in document's fingerprint is not higher than N/w. When we refer to a single document or record in this paper, we assume eight pages material. A document of eight pages consists of 40 000 characters for example. For the window size of five, we assume approximately 8 192 hashes per document in this work.

5.1 Fingerprint Generation

For the fingerprint computation on a GPGPU, the documents were transferred from the host to the global device memory first. Then, each block's thread copies a part of the document to the block shared memory. The block's threads compute hash values of the documents n-grams. Selected threads handle writing results to the global memory. Text data is uniformly allocated to the threads. A computed hash position in the text (pos) is directly linked to the working global thread identifier ($thId$), i.e. $thId = pos$ **mod** $nbrOfTh$, where $nbrOfThreads$ is the total number of all threads.

The experiment was conducted for an n-gram size that suits detection of similarities between obfuscated text, i.e. $k = 4$. Table 2 shows that a GPGPU outperforms a CPU up to 14 times for an experiment with 60 000 documents. The speedup was calculated as the CPU execution time divided by the GPU execution time. The increase over 10 000 in the number of documents has no impact on the acceleration. It should be noted that a fingerprint is generated once, and it can be kept in the database for subsequent use in most cases. It is reused many times during the fingerprints comparison phase. Increasing the number of documents has a significant effect on the GPGPU performance, which allows for massive parallel computations. For a CPU (single core), computation time is proportional to the number of documents. The average computation time is roughly 1.4 μs per document. The algorithm can be easily parallelized in many CPU cores, where each core generates a fingerprint for a different record.

M	GPU [ms]	1-Core CPU [ms]	GPU Speedup
2 000	2.8	30.7	10.8
8 000	9.9	147.8	14.8
30 000	36.1	472.1	13.0
60 000	71.0	970.8	13.6

Table 2. The GPU and CPU results for an n-gram generation. M is a number of documents; the window size is four

5.2 Direct Documents Comparison

Documents are compared one to another in the direct document's comparison. This text-to-text comparison scheme is necessary for obfuscated document because it allows the calculation of the percentage of matching hashes for selected documents. However, such comparison is a very time-consuming operation. Different to the hash generation, the hash comparison in the entire database needs to be performed for every single document addition. Therefore, in the case when generated hash reusing is possible, hash generation time has the insignificant effect on the total computation time.

Computation time, which is required for hash comparison can be significantly reduced if two assumptions are introduced. First, the examined text is compared with the aggregated fingerprints of the whole database, and the second, fingerprint's hashes are first sorted. This sorting can be a part of the fingerprint generation process, and it is performed only once per document.

To emphasize a CPU's and GPGPU's unique platform features, the authors implemented two versions of the fingerprint comparison operation. Those are the plain method and sorting-base method. Comparison times of the single document with databases of different sizes, when fingerprints' hashes are unsorted, are given in Table 3. Corresponding execution times for the unsorted fingerprints' hashes can be seen in Table 4.

M	GPU [s]	1-Core CPU [s]	GPU Speedup
1 024	14	210	14.5
8 192	114	1 648	14.7

Table 3. Comparison time for unsorted fingerprints. M is a number of documents.

M	GPU [ms]	1-Core CPU [ms]	12-Core CPU [ms]	CPU Speedup
8 192	106	98	15	7.06
16 384	217	244	29	7.48
32 768	406	440	59	6.88
49 152	575	650	100	5.75
65 536	762	900	135	5.64
131 072	1 550	1 720	280	5.53

Table 4. Comparison time for sorted fingerprints. M is the number of documents.

Assuming that each document is represented by h hashes. Comparing one document with a database containing M documents requires an order of $O(M * h^2)$ hash comparisons for unsorted and only $O(M * h)$ for sorted hashes. Assuming that the number of hashes $h = 8\,192$, the 1-core CPU computation time for sorted hashes is roughly 16 000 times smaller (compare results for 8 192 documents in Tables 3 and 4). Additionally, as a CPU can benefit from parallel execution here, an OpenMP implementation for 12 core was tested as well. Unfortunately, for the sorted hashes,

the GPGPU implementation did not introduce any speedup in comparison to the multi-core CPU solution. The above results were gathered as average values from ten experiments (standard deviation less than a few percent).

If the hash value distribution was different, i.e., the data range was narrower, it would be possible to employ the histogram computation and intersection calculation. This method could be easily parallelized on a GPU. Unfortunately, the input data is unsuitable for the histogram-based solution.

5.3 Reverse Index Fingerprint Comparison

The n-gram comparison, which adopts the threshold value is computation intensive, and therefore it should be adopted only if it is unavoidable. This could be the case when obfuscated text is expected, or detailed fine grain document comparison is desired. A Threshold-based n-gram text comparison would be necessary for the refinement stage of an anti-plagiarism system for example. The method that is based on the *no-threshold* experiment, which was presented in Section 4.2, can be considered for a large plagiarism detection system. The method presented in Section 4.2 is suitable to implement the data retrieval stage only. An extrapolation of the result for the database that contains 131 072 pages (Table 4) leads to the conclusion that comparison of one document to a database containing 600 million documents ($M = 600\,000\,000$) requires roughly 1 280 seconds if 12 CPUs are used ($1\,280\,\mathrm{s} \approx 0.28\,\mathrm{s} \times 600\,000\,000/131\,072$). Consequently for large databases a reverse index comparison is proposed. The following solution is suitable to detect exact text similarities, i.e. documents that share few large n-grams.

The idea of reverse index comparison is to search similarities in a new document not with respect to each database's record separately, but to the whole database simultaneously. A custom reverse-index-like data structure was introduced for this purpose. A hash table with a linked list was initially selected [19], but better results are obtained for a hash table with a sorted dynamic array, which is presented in Figure 9.

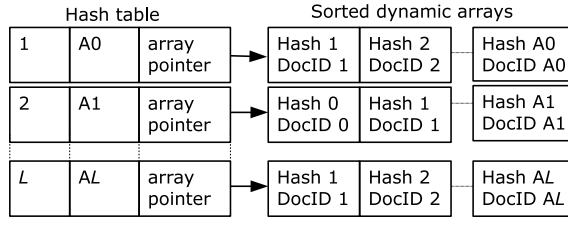


Figure 9. A custom data structure for reverse indexing

Dynamic arrays allow for sequential memory access and require less memory because there is no need to use pointers that point to the next element. As generated hashes, i.e., elements of the fingerprint are uniformly distributed, only the most

significant bits (*MSBs*) of the hash index the hash table. Besides, these *MSBs* may not be stored in a dynamic array as every hash entry in this array has the same *MSBs*. Instead of *MSBs*, a document identification number (*docID*) is stored in the dynamic array. There are enough bits available because the size of the hash table L is higher than the number of documents M. The main drawback of employing sorted dynamic array (if compared to linked list) is that the insertion of a new element requires relatively large data movements. Therefore, different average array sizes A and sizes of the hash table L were tested (see Table 5). It should be noted that $A * L = M * h$, where: M is the number of documents, and h is an average number of hashes per document.

As it can be seen in Table 5, the optimum array size A is 64 to 256.

It should be noted that the larger the A, the less memory allocation operation is necessary. It is necessary to allocate and move data to the new bigger and delete the old memory to enlarge the hash array. Simultaneously, for large A ($A > 16$) more data movement is required to insert a new element into the array as the array structure is sorted. To be more specific, on average $A/2$ elements should be shifted one position to the right to insert a new element.

Consequently, time to perform data movement dominates for larger A, and dynamic memory allocation increases the computation time for smaller A values. An array size is increased by the quantum of $A/8$ when a bigger array is required to store new elements to decrease the number of *allocation* and *de-allocation* functions calls. Thus, the allocation function is called roughly $8 * L$ times during the entire structure creation process.

A	Generation [s]	Comparison per Doc. [ms]
16	18.3	1.14
64	17.1	1.53
256	17.0	1.46
1 024	26.6	1.86

Table 5. Database generation and the document comparison times for different average dynamic array size A, $M = 8\,\text{k}$, $h = 8\,\text{k}$

The main advantage of the sorted array, which contains all fingerprints' hashes is that the search time is reduced. The most commonly used method for searching an element in a sorted array is a binary (half-interval) search. The uniform distribution of hashes is exploited in the proposed algorithm. Consequently, as a searched hash and array size are known, the place where the hash should be expected is roughly evaluated and then only neighboring hashes compared. Table 6 gives the generation and comparison times for the different number of documents M. In Tables 5 and 6 the generation time is the time required to construct the database from the start (fingerprints generation is not included). The comparison time measurements were performed for a set of 1 000 input documents, and the values that are given in the tables corresponds to the time of a one document analysis. It can be

seen from Table 6 that the generation time per document grows insignificantly with the number of records M. The comparison time is roughly constant.

When comparing Table 4 with Table 6, a conclusion can be drawn. The direct comparison should be used when a few documents from the database are compared only, or text obfuscation is expected. In the case, when material added to the database can be compared with the aggregated fingerprint data and no additional *join* operation is necessary, the reverse index comparison works much faster. If it is required to calculate the percentage of shared fingerprints' hashes of two distinctive documents, a *join* operation over *docID* is necessary. As we know short n-grams should be used for obfuscated material. If it is possible to tell the similarity by a single n-gram matches no further *join* operation is necessary.

M	Generation [s]	Gen. per Doc. [ms]	Comparision per Doc. [ms]
1 k	1.51	1.47	1.13
4 k	8.22	2.00	1.52
16 k	38	2.32	1.59
64 k	190	2.90	1.30
128 k	435	3.32	1.31

Table 6. Database generation and the document comparison times for a different number of documents M, $h = 8\,\text{k}$, $A = 64$

The reverse index requires similar memory resources as a direct method. All hashes in the table have the same document *docID* in the direct method. In the reverse index method, most significant bits of hashes are the same. Therefore, they do not need to be stored, and this space can be used to store documents' *docID*. Lower memory usage can be obtained thanks to the hashes compression. The simplest method is to use differential coding for sorted hashes. For a larger number of documents, the database is too large to be stored in the operation memory. Therefore, disk storage is required. In that case, the comparison time is determined mostly by disk access time. Consequently, the GPGPU implementation is not considered at this moment as the reverse index algorithm is sequential and requires large memory transfers rather than computation power.

6 CONCLUSIONS AND FUTURE WORK

The experiments conducted by the authors showed that there is a certain threshold of n-gram and window values which yield the best results for obfuscated and unobfuscated material. The presented n-gram-based approach is mostly effective for unobfuscated duplication, i.e. one-to-one 'copy and paste' operation. Duplication detection for obfuscated material is more challenging than for unobfuscated material. Nevertheless, it was possible to adjust the n-gram and window size in the course of experiments to get satisfactory results. Unfortunately, the detection results were worse than for unobfuscated documents, which is reflected in the lower *precision* value.

GPGPU acceleration results for fingerprint generation are high. However, it is worth noting that the comparisons were conducted against a single CPU core and with no SSE operations implemented.

The source files of the modules are available at [20]. The system is currently at its early development stage and has been partially implemented. Nevertheless, the authors described the n-gram-based algorithms along with the results of the preliminary detection quality measurements. The authors are going to implement the parts of the algorithm as hardware modules in the future [21, 22]. The most suitable for hardware implementation are the modules that perform massive parallel pattern matching such as fingerprint comparison [23, 24, 25, 26, 27].

Acknowledgments

The work presented in this paper was financed by Polish National Center for Research and Development through Synat *(SP/I/1/77065/10)*, PLGrid Plus *(POIG.02.03.00-00-096/10)* and PLGrid Core *(POIG.02.03.00-12-137/13)* the research programs.

REFERENCES

[1] IDC Predicts 2012 Will Be the Year of Mobile and Cloud Platform Wars as IT Vendors Vie for Leadership While the Industry Redefines Itself. `http://www.businesswire.com/news/home/20111201005201/en/IDC-Predicts-2012-Year-Mobile-Cloud-Platform` [access: 16.01.2014].

[2] Hilbert, M.—López, P.: The Worlds Technological Capacity to Store. Science, Vol. 332, 2011, No. 6025, pp. 60–65.

[3] Schleimer, S.—Wilkerson, D. S.—Aiken, A.: Winnowing: Local Algorithms for Document Fingerprinting. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '03), 2003, pp. 76–85, doi: 10.1145/872757.872770.

[4] Miller, E.—Shen, D.—Liu, J.—Nicholas, Ch.—Chen, T.: Techniques for Gigabyte-Scale N-Gram Based Information Retrieval on Personal Computers. Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '99). CSREA Press, 1999, pp. 1410–1416.

[5] Maurer, H.—Kappe, F.—Zaka, B.: Plagiarism – A Survey. Journal of Universal Computer Science, Vol. 12, 2006, No. 8, pp. 1050–1084.

[6] Sheard, J.—Dick, M.: Directions and Dimensions in Managing Cheating and Plagiarism of IT Students. Proceedings of the Fourteenth Australasian Computing Education Conference, Vol. 123, 2012, pp. 177–186.

[7] Potthast, M.—Stein, B.—Eiselt, A.—Barrón-Cedeño, A.—Rosso, P.: Overview of the 1st International Competition on Plagiarism Detection. In: Stein, B., Rosso, P., Stamatatos, E., Koppel, M., Agirre, E. (Eds.): SEPLN 09 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN '09), 2009.

[8] Kiela, D.—Clark, S.: A Systematic Study of Semantic Vector Space Model Parameters. Proceedings of the 2nd Workshop on Continuous Vector Space Models and Their Compositionality (CVSC) at EACL 2014, 2014, pp. 21–30, doi: 10.3115/v1/W14-1503.

[9] Lashkari, A. H.—Mahdavi, F.—Ghomi, V.: A Boolean Model in Information Retrieval for Search Engines. Proceedings of the International Conference on Information Management and Engineering (ICIME 2009), 2009, pp. 385–389, doi: 10.1109/ICIME.2009.101.

[10] Carrillo, M.—Eliasmith, Ch.—López-López, A.: Combining Text Vector Representations for Information Retrieval. Proceedings of the 12th International Conference on Text, Speech and Dialogue (TSD '09), Springer-Verlag, Berlin, Heidelberg, 2009, pp. 24–31, doi: 10.1007/978-3-642-04208-9_7.

[11] Grozea, C.—Popescu, M.: Encoplot-Performance in the Second International Plagiarism Detection Challenge – Lab Report for PAN at CLEF 2010.

[12] Potthast, M.—Stein, B.—Eiselt, A.—Barrón-Cedeño, A.—Rosso, P.: Overview of the 2nd International Competition on Plagiarism Detection. In: Braschler, M., Harman, D., Pianta, E. (Eds.): Notebook Papers of CLEF 10 Labs and Workshops, 2010.

[13] Potthast, M.—Stein, B.—Eiselt, A.—Barrón-Cedeño, A.—Rosso, P.: Overview of the 3rd International Competition on Plagiarism Detection. In: Petras, V., Forner, P., Clough, P. D. (Eds.): Notebook Papers of CLEF 11 Labs and Workshops, 2011.

[14] Potthast, M.—Gollub, T.—Hagen, M.—Grassegger, J.—Kiesel, J.—Michel, M.—Oberländer, A.—Tippmann, M.—Barrón-Cedeño, A.—Gupta, P.—Rosso, P.—Stein, B.: Overview of the 4th International Competition on Plagiarism Detection. In: Forner, P., Karlgren, J., Womser-Hacker, Ch. (Eds.): CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 2012.

[15] Potthast, M.—Hagen, M.—Gollub, T.—Tippmann, M.—Kiesel, J.—Rosso, P.—Stamatatos, E.—Stein, B.: Overview of the 5th International Competition on Plagiarism Detection. In: Forner, P., Navigli, R., Tufis, D. (Eds.): Working Notes Papers of the CLEF, 2013.

[16] Cavnar, W. B.—Trenkle, J. M.: N-Gram-Based Text Categorization. Proceedings of 3rd Annual Symposium on Document Analysis and Information Retrieval, (SDAIR-94), 1994, pp. 161–175.

[17] Heintze, N.: Scalable Document Fingerprinting. Proceedings Usenix Workshop on Electronic Commerce, 1996.

[18] PAN Competition – `http://pan.webis.de/` [Accessed: 10.02.2014].

[19] Cormen, T.—Leiserson, C.—Rivest R.: Introduction to Algorithms. MIT, 1994.

[20] Piertoń, M.—Jamro, E.—Żurek D.: The Experiments Source Files. `https://git.plgrid.pl/users/plgwielgosz/repos/evaluation_and_implementation_of_n-gram-based_algorithm_for_fast_text_comparison`, 2015.

[21] Wielgosz, M.—Mazur, G.—Makowski, M.—Jamro, E.—Russek, P.—Wiatr, K.: Analysis of the Basic Implementation Aspects of Hardware-Accelerated

Density Functional Theory Calculations. Computing and Informatics, Vol. 29, 2010, No. 6, pp. 989–1000.

[22] Jamro, E.—Russek, P.—Dabrowska-Boruch, A.—Wielgosz, M.—Wiatr, K.: The Implementation of the Customized, Parallel Architecture for a Fast Word-Match Program. Computer System Science and Engineering, Vol. 26, 2011, pp. 285–292.

[23] Pietron, M.–Wielgosz, M.—Zurek, D.—Jamro, E.—Wiatr, K.: Comparison of GPU and FPGA Implementation of SVM Algorithm for Fast Image Segmentation. Proceedings of Architecture of Computing Systems: 26th International Conference. Springer-Verlag, Lecture Notes in Computer Science, Vol. 7767, 2013, pp. 292–302.

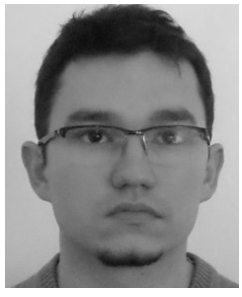
[24] Wielgosz, M.—Panggabean, M.—Wang, J.—Rønningen, L. A.: An FPGA-Based Platform for a Network Architecture with Delay Guarantee. Journal of Circuits Systems and Computers, Vol. 22, 2013, No. 6, pp. 1350045-1–1350045-20, doi: 10.1142/S021812661350045X.

[25] Kryjak, T.—Gorgon, M.: Pipeline Implementation of Peer Group Filtering in FPGA. Computing and Informatics, Vol. 31, 2013, No. 4, pp. 727–741.

[26] Zurek, D.—Pietroń, M.—Wielgosz, M.—Wiatr, K.: Comparison of Hybrid Sorting Algorithms Implemented on Different Parallel Hardware Platforms. Computer Science, Vol. 14, 2013, No. 4, pp. 679–691.

[27] Kuta, M.—Kitowski, J.: Comparison of Latent Semantic Analysis and Probabilistic Latent Semantic Analysis for Documents Clustering. Computing and Informatics, Vol. 33, 2014, No. 3, pp. 652–666.

Maciej Wielgosz received his Engineering degree and his Ph.D. degree (with honors) in electronics from the AGH University of Science and Technology, Krakow, Poland, in 2006 and 2010, respectively. He is currently Assistant Professor in the Department of Electronics, AGH and works in the Academic Computing Centre CYFRONET. His main areas of research interest are machine learning, image and natural langue processing, and hardware architectures for artificial intelligence. He has published over 80 technical papers.



Paweł Szczepka received his B.Sc. Eng. and M.Sc. degrees in electronic engineering in 2012 and 2013, respectively, from the AGH University of Science and Technology, Kraków, Poland. Currently he works at FORTRESS Gaming Technologies. His research interests include data mining and natural language processing.

Paweł Russek received his Ph.D. degree in electronics from the AGH University of Science and Technology in Krakow, Poland in 2002. He is currently Assistant Professor at the AGH-UST at Faculty of Computer Science, Electronics and Telecommunications, and also, he works in the Academic Computer Centre Cyfronet AGH as a manager of the Computing Acceleration Group. His interests focus on novel computer architectures, hardware accelerators, and custom computing processors. He is an author and co-author of over 100 publications in the area of accelerated computing using GPGPU and FPGA-enabled hybrid systems. His research has been focused on new algorithms suited for efficient processing by custom computing architectures.

Ernest Jamro received his M.Sc. degree in electronic engineering from the AGH University of Science and Technology (AGH-UST), Krakow, Poland in 1996, M.Phil. degree from the University of Huddersfield (U.K.) in 1997; his Ph.D. and habilitation (Dr.Hab.) degrees from the AGH-UST in 2001 and 2014, respectively. He is currently Assistant Professor in the Department of Electronics, AGH-UST. His research interests include reconfigurable hardware (especially Field Programmable Gate Arays – FPGAs), reconfigurable computing systems, system on chip, and artificial intelligence.

Kazimierz Wiatr received his M.Sc. and Ph.D. degrees in electrical engineering from the AGH University of Science and Technology, Krakow, Poland, in 1980 and 1987, respectively, and the Dr.Hab. degree in electronics from the University of Technology of Łódź in 1999. He is Full Professor since 2002. His research interests include design and performance of dedicated hardware structures and reconfigurable processors employing FPGAs for acceleration computing. Currently he is Director of the Academic Computing Centre CYFORNET AGH.

Marcin Pietroń received his M.Sc. degree in electronic engineering and in computer science in 2003 and his Ph.D. degree in 2013 from the AGH University of Science and Technology, Krakow, Poland. He currently works in the Academic Computing Centre CYFRONET AGH and at the University of Science and Technology. His research interests include parallel computing, automatic parallelization and machine learning.

Dominik Żurek received his B.Sc. Eng. and M.Sc. degrees in electronic engineering in 2011 and 2012, respectively, from the AGH University of Science and Technology, Kraków, Poland. Currently he is a Ph.D. student. His research interests include parallel computing, automatic parallelization and data mining.