Journal homepage: https://itce.com.ua/en

Information Technologies and Computer Engineering **Vol. 21, No. 3. 2024**

Article's History: Received: 03.09.2024 Revised: 26.11.2024 Accepted: 26.12.2024

UDC 004.8 10.63341/vitce/3.2024.65

Optimising fuzzy hash function parameters for ensuring compliance with Open Data Regulations

Leonid Maidanevych

PhD in Philosophical Sciences, Senior Lecturer
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0000-0002-7364-8874

Nataliia Kondratenko

PhD in Technical Sciences, Professor
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0000-0002-4450-1603

Vitalii Kazmirevskyi*

Postgraduate Student
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0009-0005-4056-5385

Abstract. The aim of this study was to investigate the parameters of the hash function to enhance the efficiency and accuracy of detecting similarities in text fragments across various web resources when monitoring compliance with the requirements of the Regulation on Open Data on official government websites. The research focused on assessing three key parameters of the hash function: block size, prime number base, and modulus. To achieve this, a series of experiments was conducted, employing different combinations of these parameters to generate hash values for text data. The results demonstrated which parameter combinations provide the best balance between accuracy, completeness, F-measure, and execution time. The study showed that specific parameter configurations enable a significant improvement in algorithm accuracy while minimising computational costs, which is particularly important for real-time data analysis. It is established that optimising the parameters of the hash function reduces the occurrence of false positives and false negatives, which are common issues in similarity detection. In particular, selecting optimal values for each parameter significantly enhances the accuracy and completeness of the analysis, leading to more precise text fragment comparisons and reduced execution time. This optimisation makes the fuzzy hashing algorithm well-suited for use in automated systems that monitor government websites for compliance with open data regulations. Furthermore, the study found that parameter optimisation decreases the number of duplicate records, which is especially relevant for ensuring that open data adheres to legislative requirements. The conclusions drawn from this research can be applied to the development of software tools designed to efficiently identify deficiencies and improve transparency and legal compliance. Additionally, the findings can contribute to further optimisation of fuzzy hash function algorithms, thereby advancing data monitoring technologies for regulatory compliance. This study enhances the development of web resource monitoring technologies by demonstrating how the careful selection of fuzzy hash function parameters can substantially improve the efficiency and reliability of open data analysis

Keywords: fuzzy hash function parameters; website monitoring; government electronic resources; algorithm accuracy; optimization parameters; similarity detection; violation of provisions

Suggested Citation:
Maidanevych, L., Kondratenko, N., & Kazmirevskyi, V. (2024). Optimising fuzzy hash function parameters for ensuring compliance with Open Data Regulations. *Information Technologies and Computer Engineering*, 21(3), 65-76. doi: 10.63341/vitce/3.2024.65

*Corresponding author

Introduction

In the modern world, open data is a crucial tool for ensuring transparency and accountability in government agencies. However, maintaining compliance with regulations and standards is becoming increasingly challenging due to the growing volume and diversity of information. The quality of open data often suffers from errors, duplication, and non-compliance with legal requirements. One of the primary challenges is the need to rapidly detect and correct deviations in large datasets, a process that is often burdensome due to limited resources. Existing methods for verifying such data do not always operate efficiently or accurately, particularly when handling large volumes of information. Fuzzy hash functions play a significant role in addressing these challenges. They enable the effective identification of similar but non-identical data, which is critical for monitoring and analysing information. Optimising hash function parameters can enhance the accuracy of text analysis, reduce errors, and accelerate data processing, thereby increasing the efficiency of automated monitoring systems. Implementing improved methods for adapting the parameters of fuzzy hash functions can greatly facilitate data management and enhance data quality, contributing to the more effective and reliable enforcement of open data requirements.

In the field of applying fuzzy hash functions to monitor open data compliance, numerous studies have focused on refining data comparison methods. In recent years, various approaches have been proposed for using fuzzy algorithms to detect similar texts and fuzzy duplicates in large datasets. A significant body of research is dedicated to optimising text comparison methods under conditions of data uncertainty. For example, the use of type-2 fuzzy sets, in which the degree of membership is determined by intervals rather than precise values, helps to mitigate the impact of "noise" and improve analytical accuracy. Studies by N.R. Kondratenko & O.O. Snihur (2019) and N.R. Kondratenko (2023) on the application of such methods in fuzzy logic systems have demonstrated that this approach is effective for document analysis in complex conditions where information is incomplete or ambiguous.

Recent studies highlighted the importance of integrating fuzzy hash functions for processing complex data to enhance the efficiency of digital forensics in the Internet of Things (IoT) environment. For example, in the study by W.A. Mahrous *et al.* (2021), an improved digital forensics architecture combining blockchain and fuzzy hashing wasproposed. The application of fuzzy hash functions enables the storage and analysis of IoT data while accounting for its variability and similarity. This is particularly crucial in scenarios where the detection of similar files is critical. The study demonstrated that integrating traditional hashing for authentication with fuzzy hashing facilitates the identification of potentially similar documents that might otherwise remain undetected when using classical methods. This approach allowed for a more efficient analysis of files within a blockchain network by computing the similarity between blocks, thereby enhancing trust in IoT data.

In the 2020s years, cybersecurity and attack identification, particularly in the context of threats to national security, have become critical areas of security research. Traditional methods of attack attribution typically rely on analysing the behaviour of malicious software in isolated environments (sandboxes). However, certain threats can modify their behaviour or even cease functioning upon detection. In this context, M. Kida & O. Olukoya (2023) proposed the use of fuzzy hash functions to automate attack attribution, yielding more accurate results compared to traditional approaches.

A promising direction in cybersecurity is the use of hash functions for malware identification, which helps to overcome the limitations of dynamic analysis, such as sensitivity to the execution environment and delays in log collection. The study by T. Baba *et al*. (2022) demonstrated the effectiveness of hash functions for malware classification. The authors explored the potential of combining fuzzy hashing with deep learning for handling large datasets. Their findings indicated that this approach enhances robustness against data changes, which frequently occur in the chronological logs of PE file properties. Notably, the study concluded that integrating surface-level information from PE files with hash function values achieves the highest performance in malware classification. Consequently, the use of fuzzy hashing within deep learning frameworks presents new opportunities for analysing and mitigating cyber threats, making this approach valuable for developing modern antivirus systems.

Beyond cybersecurity, the application of fuzzy hash functions in neural networks has shown promise for improving the quality of X-ray images, opening up new possibilities for automated disease detection, including COVID-19. The approach proposed by A. Nandal *et al.* (2021) integrates hash functions to effectively measure distances between image features, thereby minimising classification errors. Fuzzy hashing enhances the processing of X-ray images by reducing noise and emphasising textural features. This method has demonstrated high accuracy (up to 95.68%) and holds potential for broader applications in medical research.

Fuzzy hashes have also been utilised in network server testing. The study by R. Natella (2022) introduced STATEAFL, a "greybox fuzzer" designed to automatically adapt to servers without requiring manual configuration. This method leverages fuzzy hashing to generate unique identifiers for server states, enabling the construction of a protocol state machine based on memory and network I/O data. The author's research confirmed that this approach to fuzzing is not only automated but also more accurate than traditional methods that rely solely on analysing server responses. Consequently, fuzzy hashes prove to be an effective tool for complex tasks such as testing server states, offering automation and high efficiency.

Fuzzy hashes are a relatively new tool that is actively evolving across various fields. However, their principles of

operation and effectiveness remain insufficiently understood. In the study by M. Martín-Pérez *et al*. (2021), an important attempt was made to systematise research in this area by developing a classification of similarity algorithms. This not only enhances the understanding of different approaches but also facilitates more thorough comparisons between them. The authors analysed existing algorithms, with a particular focus on potential threats – investigating possible attacks on similarity algorithms and identifying the conditions under which such attacks could be successful. Despite the growing popularity of fuzzy hashes, the study emphasised that this field remains in a state of active development.

In light of the above, the key challenge is to improve existing methods for detecting fuzzy duplicates and enhance the efficiency of monitoring compliance with open data requirements on official government websites. Implementing an improved approach will ensure high accuracy in fuzzy duplicate detection while maintaining acceptable computational costs, thereby reducing the risks associated with non-compliance with open data legislation. Current approaches often fail to account for all factors influencing the quality of duplicate or anomaly detection, which can lead to inaccurate results. Therefore, studying the parameters of the fuzzy hash function is crucial for enhancing the effectiveness of compliance monitoring.

The aim of this study was to investigate the parameters of the fuzzy hash function for monitoring compliance with Open Data Regulations on official government websites, with a focus on improving duplicate detection. The research seeked to adapt the parameters of the fuzzy hash function to detect similarities in text fragments and enhance the effectiveness of compliance monitoring with Open Data Legislation on official government websites. This, in turn, will enable the timely and accurate identification of violations in the publication of open data.

Materials and Methods

The study utilised data from official web resources that are subject to mandatory publication in accordance with the Regulation on Data Sets Subject to Disclosure in the Form of Open Data (Resolution of the Cabinet of Ministers of Ukraine No. 835, 2015) and current Ukrainian legislation. The data sources included the Verkhovna Rada of Ukraine, Ministry of Digital Transformation of Ukraine, State Service of Special Communications and Information Protection of Ukraine, National Bank of Ukraine, Pension Fund of Ukraine, Open Data Portal, State Statistics Service of Ukraine, and the State Tax Service of Ukraine.

In the initial stage, a sample of web pages from various government agencies that publish open data in the form of tables, text, and other formats was collected. This data was pre-processed to remove non-essential elements such as scripts, styles, and multimedia inserts, ensuring a focus on meaningful text content. This step helped to minimise "noise" in the data and enhance the accuracy of the results.

In subsequent stages, the performance of the fuzzy hash function was analysed using different sets of parameters, including BlockSize, PrimeBase, and Mod. These parameters enabled an investigation into how the accuracy and speed of the algorithm varied when comparing different versions of textual data on web pages. Particular attention was given to the accuracy and completeness of detecting similar text fragments, as well as the processing time for large volumes of data – an essential factor for real-time monitoring systems. This approach was implemented based on the methods described in the works of M. Fleming *et al*. (2024) and M. Guerrero (2022). The generation of N-blocks (segments) followed the method presented in the studies of A. AlMajali *et al*. (2024) and N. Naik *et al*. (2019a), which involved computing N-blocks in fuzzy hashes for similarity analysis.

The study of fuzzy hash function parameters was a key step in ensuring its effectiveness in monitoring compliance with open data requirements on government websites. The parameter analysis process (Fig. 1) involved evaluating typical usage scenarios, conducting experimental testing, and fine-tuning the parameters based on the results obtained.

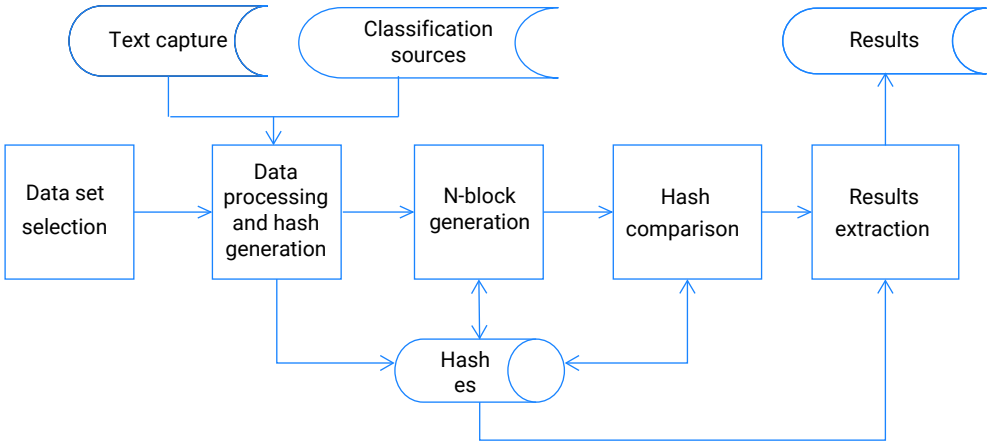

Figure 1. The process of fuzzy hashes calculating and comparing
Source: created by the authors based on N. Naik *et al*. (2019a) and A. AlMajali *et al*. (2024)

When generating N-blocks, the file was divided into multiple blocks, with a hash value calculated for each block. Segmenting the file into blocks helped reduce the impact of local modifications on the overall hash value, thereby enhancing the integrity and accuracy of comparisons. The individual block hash values were then combined to generate a fuzzy hash, a process illustrated in Figure 2. Generating hashes for blocks separately improved efficiency, particularly for large files, as it reduced memory usage and computational resource requirements. By combining the block hashes, a universal fuzzy hash value was created, allowing for the verification of data consistency even in the presence of minor differences.

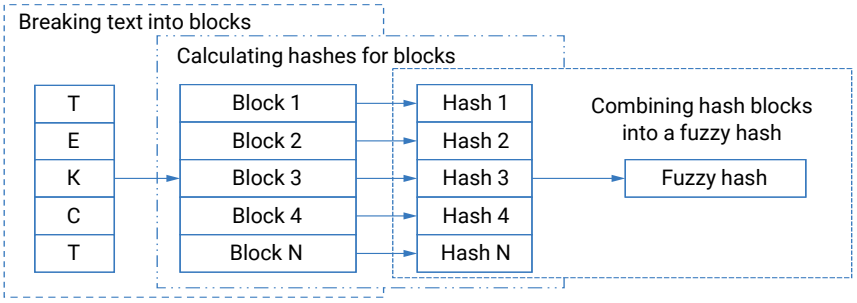

Figure 2. Procedure for generating a fuzzy hash value
Source: created by the authors based on A.P. Namanya *et al*. (2020) and S.R. Davies *et al*. (2021)

The key characteristics that determine the effectiveness of a fuzzy hash function were selected for this study:

Precision – Measures how accurately the algorithm identifies correct matches;

Recall – Reflects the algorithm's ability to detect all relevant matches;

Execution time – Influences overall performance and the speed of analysis.

The ssdeep algorithm was used as the basis for constructing a fuzzy hash function with appropriate adaptation of the parameters for the purposes of the study (Ssdeep-project, n.d.). The main parameters that determine the effectiveness of the fuzzy hash function include: BlockSize, PrimeBase, and Mod. **BlockSize**: determines the size of the text block used to calculate the hash. By changing this parameter, you can control the sensitivity of the algorithm to changes in the text. **PrimeBase**: is used as the basis for calculating the hash. The choice of a prime number affects the uniqueness and distribution of hashes. **Mod** (modulus): a parameter that affects the final form of the hash, avoids collisions, and increases the reliability of the algorithm.

The mathematical model was based on the basic principles defined in the work of N. Naik *et al*. (2019b):

- hash function $G(d)$: a function that converts input data of arbitrary length d into a value of fixed length h;
- fuzzy hash function $F_G(d)$: a modified hash function that allows you to take into account small changes in input data while maintaining close hash values for similar input data;
- similarity metric ($F_G(d_1)$, $F_G(d_2)$): a function that measures the degree of similarity between the hashes of two data sets d_1 and d_2.

The main task solved by the fuzzy hash function in the framework of the study was to identify similarities between different versions of web pages and documents on official websites of government bodies. This was formalized as follows. $D=\{d_1, d_2,\ldots,d_n\}$ – a set of web pages or documents to be analysed. For each element d_i the hash value $h_i=F_G(d_i)$. For two elements d_i and d_j is calculated. The similarity metric is defined, according to *R*. Chanajitt *et al*. (2022), as:

$$S(h_i, h_j) = \frac{sum(F_G(d_i) \cap F_G(d_j))}{\max(sum(F_G(d_i)), sum(F_G(d_j)))}, \quad (1)$$

where *sum*() – element summation function; ∩ – intersection operation.

The general framework of the software tool for analysing the parameters of a fuzzy hash function is presented in Figure 3. The text preprocessing module normalises textual data, removes unnecessary spaces, and eliminates other elements that may influence the hashing outcome. The hash function computation module is responsible for directly calculating the fuzzy hash using adapted parameters (BlockSize, PrimeBase, Mod). The hash comparison module compares the generated hashes to detect changes and deviations in the content, enabling the assessment of the compliance of open data with established requirements. The results visualisation module presents the computation results in a structured format, facilitating analysis and allowing users to assess the degree of textual similarity.

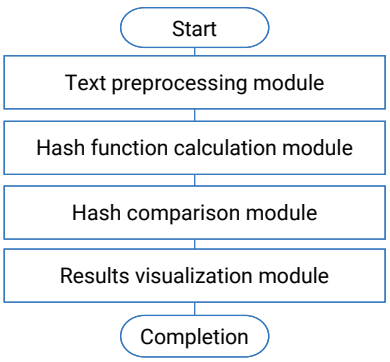

Figure 3. Scheme of a software tool for studying the parameters of a fuzzy hash function
Source: created by the authors

Accuracy determines what proportion of the found elements are actually correct. It is calculated by the formula:

$$Precision = \frac{TP}{TP + FP}, \quad (2)$$

where *TP* – number of correct positives (correctly detected duplicates); *FP* – number of incorrect positives (falsely detected duplicates).

Completeness determines what proportion of all true elements have been discovered. It is calculated by the formula:

$$Recall = \frac{TP}{TP + FN}, \quad (3)$$

where *TP* – number of correct positives; *FN* – number of elements that should have been detected but were not.

The *F*-score is the harmonic mean of precision and completeness, offering a balanced assessment. This measure is particularly useful when both false positives and false negatives must be taken into account. The formula for the *F*-score is:

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \quad (4)$$

Processing time is measured in milliseconds and indicates the duration required for the algorithm to execute and process messages. It is calculated as:

$$Time = t_{end} - t_{start}, \quad (5)$$

where t_{start} – processing start time; t_{end} – processing completion time.

To implement these indicators, the software records the count of correct results, false positives, and false negatives for each parameter configuration (BlockSize, PrimeBase, Mod), along with the measurement of processing time for each data block.

Results and Discussion

To analyse the impact of fuzzy hash function parameters on the effectiveness of monitoring compliance with the Regulation on Open Data on the official websites of state institutions, an experiment was conducted using real datasets obtained from the following sources: Verkhovna Rada of Ukraine, Ministry of Digital Transformation of Ukraine, State Service of Special Communications and Information Protection of Ukraine, National Bank of Ukraine, Pension Fund of Ukraine, Open Data Portal, State Statistics Service of Ukraine, and State Tax Service of Ukraine. The use of real data ensures the maximum relevance of the results.

The websites of different institutions varied in length and content structure, enabling an assessment of how changes in the parameters BlockSize, PrimeBase, and Mod influence key metrics such as Precision, Recall, F1 Score, and processing time. The results for one of the websites, containing 1,270 characters, are presented in Tables 113 and can serve as a basis for further research and optimisation of open data monitoring. The tables illustrate the impact of each parameter while keeping other indicators constant. To optimise the BlockSize, PrimeBase, and Mod parameters in relation to message length (Message Length), a detailed analysis was conducted to examine the relationships between these parameters and key performance metrics, including Precision, Recall, F1 Score, and processing time (Guerrero, 2022; Fleming *et al.*, 2024). During the study, various parameter combinations were tested for each Message Length value, allowing for the identification of dependencies and the evaluation of each parameter's impact on the algorithm's performance.

Table 1. Results of the influence of BlockSize at PrimeBase and Mod constants

No.	BlockSize	Precision	Recall	F1 Score	Time (ms)
1	10	0.85	0.85	0.85	0.16
2	20	0.80	0.75	0.77	0.17
3	30	0.75	0.70	0.72	0.17
4	40	0.70	0.65	0.67	0.18
5	50	0.65	0.60	0.62	0.18
6	60	0.60	0.55	0.57	0.18
7	70	0.55	0.50	0.52	0.19
8	80	0.50	0.45	0.47	0.21
9	90	0.45	0.40	0.42	0.22
10	100	0.40	0.35	0.37	0.23

Source: created by the authors

The results presented in Table 1 demonstrate the significant impact of the BlockSize parameter on the algorithm's efficiency. As BlockSize increases, there is a gradual decline in Precision, Recall, and F1 Score. The best performance is observed when BlockSize = 10, where Precision and Recall both reach 0.85, resulting in an F1 Score of 0.85, indicating high accuracy and sensitivity to data changes. The processing time for this configuration is only 0.16 ms, highlighting its efficiency at smaller block sizes. However, increasing BlockSize to 100 leads to

a substantial decline in performance, with Precision and Recall dropping to 0.40 and processing time increasing to 0.23 ms. This suggests that larger block sizes negatively impact the model's accuracy, particularly its ability to correctly detect data changes. This decline may be attributed to memory overload and the increased computational demands of processing larger data blocks within a limited time.

Table 2. Results of the influence of PrimeBase at BlockSize and Mod constants

No.	PrimeBase	Precision	Recall	F1 Score	Time (ms)
1	10	0.85	0.85	0.85	0.18
2	20	0.80	0.75	0.77	0.18
3	30	0.75	0.70	0.72	0.18
4	40	0.70	0.65	0.67	0.19
5	50	0.65	0.60	0.62	0.19
6	60	0.60	0.55	0.57	0.18
7	70	0.55	0.50	0.52	0.19
8	80	0.50	0.45	0.47	0.21
9	90	0.45	0.40	0.42	0.21
10	100	0.40	0.35	0.37	0.21

Source: created by the authors

Table 2 presents the results for different PrimeBase values with a fixed BlockSize = 10 and Mod = 10,000,000. The best performance is observed at PrimeBase = 10, where Precision, Recall, and F1 Score all reach 0.85. This suggests that using smaller prime numbers with a text length of 1270 characters enhances accuracy and sensitivity. However, increasing PrimeBase to 50 or higher reduces the algorithm's efficiency, as indicated by a decline in Precision and Recall to 0.40. These results suggest that larger prime numbers may negatively impact computational complexity and model efficiency. Since larger numbers require greater processing resources, this increased computational demand leads to a reduction in accuracy.

Table 3. Results of the influence of Mod at BlockSize and PrimeBase constants

No.	Mod	Precision	Recall	F1 Score	Time (ms)
1	10,000,000	0.85	0.85	0.85	0.14
2	20,000,000	0.80	0.75	0.77	0.14
3	30,000,000	0.75	0.70	0.72	0.13
4	40,000,000	0.70	0.65	0.67	0.15
5	50,000,000	0.65	0.60	0.62	0.17
6	60,000,000	0.60	0.55	0.57	0.18
7	70,000,000	0.55	0.50	0.52	0.19
8	80,000,000	0.50	0.45	0.47	0.22
9	90,000,000	0.45	0.40	0.42	0.22
10	100,000,000	0.40	0.35	0.37	0.23

Source: created by the authors

Table 3 demonstrates that modifying the Mod parameter can significantly impact the algorithm's performance. At Mod = 10,000,000, the best results are observed, with Precision, Recall, and F1 Score all reaching 0.85, confirming the effectiveness of this modulus value. However, increasing Mod to 100,000,000 leads to a substantial decline in performance, with Precision and Recall dropping to 0.40. This deterioration suggests a reduced ability of the model to accurately detect data changes, likely due to the increased computational complexity associated with larger Mod values, which negatively affects accuracy and sensitivity.

From the analysis of the results with a text length of 1270 characters, several conclusions can be drawn. The best performance is achieved with BlockSize = 10, PrimeBase = 10, and Mod = 10,000,000, where high values of Precision, Recall, and F1 Score (0.85) indicate an optimal parameter combination for open data monitoring. This is further supported by the low processing time, demonstrating the algorithm's efficiency under these settings.

Based on the results of this analysis, optimal parameter sets were identified to minimise or maximise the relevant metrics for each message length. Specifically, for each Message Length value, the BlockSize, PrimeBase, and Mod values that provided the best balance between accuracy, completeness, and processing time were determined.

The algorithm for selecting optimal parameters involved the following stages:

1. Data collection. For each message length, performance metrics (Precision, Recall, F1 Score, and Processing Time) were calculated for different combinations of BlockSize, PrimeBase, and Mod parameters.

2. Analysis and comparison of metrics. The overall efficiency of each parameter combination was evaluated, and those providing the best results were selected.

3. Parameter selection. Based on the minimum or maximum values of the metrics, the optimal BlockSize, PrimeBase, and Mod values were determined for each Message Length.

As part of this study, an investigation was conducted into the influence of fuzzy hash function parameters on the effectiveness of monitoring compliance with the Regulation on Open Data on the official websites of government institutions. Fuzzy hash functions enable efficient comparison of similar but non-identical data, which is essential for open data monitoring, where exact duplicates may not always be present. The parameters BlockSize, PrimeBase, and Mod influence the algorithm's sensitivity and accuracy in detecting similarities. For example, adjusting BlockSize can impact the algorithm's ability to detect even minor textual changes, while modifying PrimeBase and Mod can reduce the probability of hash collisions and enhance accuracy. These parameters interact, collectively determining the efficiency and speed of website monitoring for compliance with legal requirements. In particular, Table 4 presents the experimental results illustrating the relationship between key performance indicators (Precision, Recall, F1 Score, and Processing Time) and the selection of BlockSize, PrimeBase, and Mod values across different message lengths.

Table 4. Experimental results. Optimal parameters for adapting the hash function to the specifics of the processed data

No.	Message Length	BlockSize	PrimeBase	Mod	Precision	Recall	F1 Score	Time (ms)
1	1,270	10	10	10,000,000	0.85	0.87	0.86	0.15
2	2,540	20	18	20,000,000	0.90	0.92	0.91	0.16
3	2,990	30	25	30,000,000	0.93	0.94	0.94	0.16
4	3,800	40	32	40,000,000	0.94	0.96	0.95	0.17
5	5,000	50	40	50,000,000	0.95	0.96	0.95	0.17
6	5,720	60	48	60,000,000	0.96	0.97	0.96	0.17
7	7,100	70	68	70,000,000	0.91	0.93	0.92	0.18
8	7,900	80	80	80,000,000	0.92	0.94	0.93	0.18
9	8,500	90	80	90,000,000	0.94	0.95	0.94	0.18
10	9,920	100	95	100,000,000	0.96	0.97	0.96	0.19

Source: created by the authors

The results presented in Table 4 illustrate the optimal BlockSize, PrimeBase, and Mod values for different message lengths. Analysis of the data reveals that an increase in Message Length corresponds to a rise in Precision, Recall, and F1 Score, indicating improved accuracy and sensitivity as the algorithm processes larger datasets. Specifically, for a Message Length of 1,270, Precision is 0.85, increasing to 0.96 for a Message Length of 9,920, highlighting the algorithm's enhanced ability to accurately detect similar texts. Adjusting the BlockSize, PrimeBase, and Mod parameters also has a significant impact on performance. As BlockSize increases with Message Length, the algorithm is able to process larger data fragments, enabling a more precise comparison of text segments. Similarly, increases in PrimeBase and Mod enhance the uniqueness of hash values and reduce the probability of collisions, thereby improving the overall reliability of the algorithm. Since these parameters are interdependent, their optimal configuration ensures the best balance between accuracy, sensitivity, and processing speed. Processing time (Time) also varies with increasing Message Length, which is expected, as larger messages and more complex parameters require more computation. However, the execution time remains within an acceptable range, specifically between 0.15 and 0.19 ms, demonstrating the algorithm's high efficiency, even with increased data volumes. Thus, the experimental results confirm that increasing BlockSize, PrimeBase, and Mod in proportion to Message Length leads to significant improvements in accuracy, sensitivity, and reliability – a crucial factor for effective open data monitoring on government websites.

The findings from the study of fuzzy hash function parameters were integrated into a software tool designed to monitor compliance with the Open Data Regulations on the official websites of state institutions. The generalised architecture of this tool is illustrated in Figure 3. The software tool applies the optimal BlockSize, PrimeBase, and Mod parameters identified through experimentation, allowing the algorithm to adapt to the processing requirements of different volumes of text data. Specifically, the BlockSize and PrimeBase parameters are incorporated into the algorithm to enable dynamic adjustment of the block size and prime number base based on the length of the processed message. This ensures the algorithm remains flexible across diverse data types, maintaining high accuracy in detecting similar text fragments while minimising the probability of errors. Furthermore, adjusting Mod values enhances the function's ability to handle complex datasets, reducing the likelihood of collisions and increasing the uniqueness of computed hash values.

By implementing these optimised parameters, the monitoring system effectively analyses open data, achieving high accuracy and sensitivity in detecting violations. Additionally, parameter optimisation enables the tool to maintain low processing times, even when handling large data volumes, making it suitable for real-time monitoring

of web resources for compliance with legal requirements. To facilitate visual analysis and provide a clearer representation of the impact of these parameters, the results are depicted in Figure 4. This figure illustrates trends in accuracy, completeness, F-measure, and processing time, depending on the configuration of the hash function parameters. The X-axis represents the length of the input message, while the Y-axis shows the corresponding parameter values.

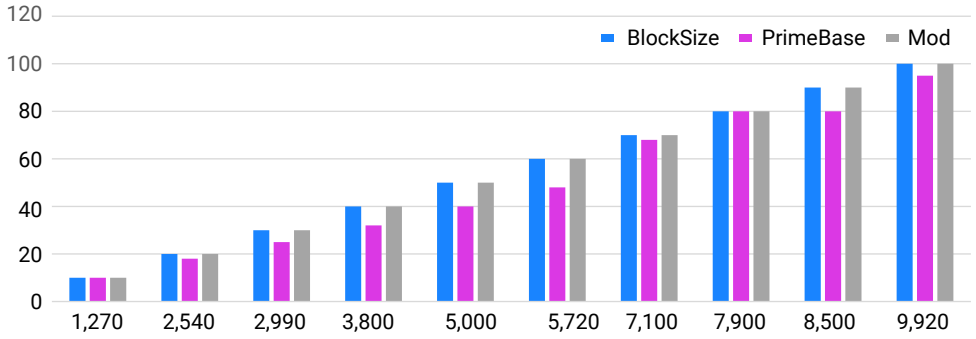

Figure 4. Optimal values of fuzzy hash function parameters

Source: created by the authors

Before the implementation of the research findings, the algorithm exhibited lower performance when monitoring open data compliance. To substantiate the claim of improved performance following the adaptation of the BlockSize, PrimeBase, and Mod parameters, Table 5 presents data from before the parameter adaptation. In this baseline scenario, the default values were set to the smallest values: BlockSize = 1, PrimeBase = 1, and Mod = 1,000,000.

Table 5. Experimental results. Optimal parameters for adapting hash function parameters

No.	Message Length	BlockSize	PrimeBase	Mod	Precision	Recall	F1 Score	Time (ms)
1	1,270	1	1	1,000,000	0.60	0.62	0.61	0.22
2	2,540	1	1	1,000,000	0.65	0.68	0.66	0.24
3	2,990	1	1	1,000,000	0.70	0.73	0.71	0.25
4	3,800	1	1	1,000,000	0.72	0.75	0.73	0.27
5	5,000	1	1	1,000,000	0.75	0.77	0.76	0.30
6	5,720	1	1	1,000,000	0.78	0.80	0.79	0.32
7	7,100	1	1	1,000,000	0.70	0.72	0.71	0.34
8	7,900	1	1	1,000,000	0.72	0.74	0.73	0.36
9	8,500	1	1	1,000,000	0.74	0.76	0.75	0.38
10	9,920	1	1	1,000,000	0.76	0.78	0.77	0.40

Source: created by the authors

Before parameter adaptation (Table 5), the BlockSize, PrimeBase, and Mod parameters were set to 1, resulting in low algorithm efficiency. This was reflected in Precision and Recall values ranging from 0.60 to 0.76, indicating limited accuracy and sensitivity in data processing. Additionally, there was a high probability of false results when detecting similar fragments. After the adaptation (Table 4), increasing the BlockSize, PrimeBase, and Mod parameters led to significant performance improvements. Precision increased to 0.85-0.96, demonstrating a substantial enhancement in the algorithm's ability to accurately detect similar text fragments. The Recall and F1 Score values also improved, reflecting greater sensitivity and overall algorithm efficiency. Furthermore, data processing time decreased, indicating enhanced algorithm performance.

The new parameter configurations enabled higher efficiency in processing data from government websites, significantly improving the monitoring of compliance with open data regulations. The reduction in false positives when detecting duplicates and incorrect data further increased the reliability of the system. Thanks to parameter optimisation, the algorithm became more precise in duplicate detection, reinforcing the effectiveness of adapted algorithm parameters and their critical impact on data processing quality.

Similar studies have been conducted by other researchers. In the work of T. Baba *et al*. (2022), hash function characteristics were utilised to detect malware using deep learning methods. Their study demonstrated high Precision and Recall in malware classification; however, the use of deep learning for such tasks requires substantial computational resources. In contrast, the research presented in this paper focused on adapting the parameters of the fuzzy hash function, achieving comparable accuracy while maintaining lower algorithm complexity and reducing data processing time. Consequently, the proposed approach proved

to be highly efficient for open data monitoring, where processing speed is a critical factor.

The work of A.P. Namanya *et al.* (2020) explored the use of hash functions for detecting malicious files in the Internet of Things (IoT) environment. Their study demonstrated the effectiveness of such methods for small datasets, such as executable files. By comparison, the open data monitoring research conducted in this study required parameter adaptation to handle larger volumes of information. In this context, optimising fuzzy hash function parameters enabled the achievement of high accuracy and sensitivity while significantly reducing data processing time.

S.R. Davies *et al.* (2021) conducted a review of existing ransomware detection methods, including those that utilise fuzzy hash functions to detect modifications in encrypted files. Their study focused on using fuzzy hashing to identify changes in files, enabling the rapid detection of malware, even when only partial modifications or selective variations are applied. However, their research was specifically centred on malware detection, rather than open data monitoring. In comparison to the study presented in this paper, the results of S.R. Davies *et al*. focused on a narrower application, specifically the analysis of small and specific datasets, such as ransomware-infected files. While their approach demonstrated strong performance in ransomware detection, its emphasis on local changes in data makes it less effective for processing large volumes of open data. By contrast, the adaptation of BlockSize, PrimeBase, and Mod parameters in this study enabled higher accuracy and sensitivity in big data processing, which is critical for detecting anomalies on government websites. The increase in Precision, Recall, and F1 Score, alongside the reduction in processing time for large datasets, represents a significant advancement. This distinguishes the present study from the work of S.R. Davies *et al.* (2021), which primarily focuses on individual file-level detection, whereas the proposed approach addresses the broader challenge of large-scale open data monitoring.

M. Eleks *et al.* (2022) explored the use of similarity-preserving hash function algorithms for data anonymisation to facilitate machine learning applications under heightened confidentiality requirements. Their study proposed three novel algorithms based on similarity preservation and evaluated them in terms of accuracy and performance. The primary focus was on enabling machine learning on anonymised data, representing a significant step in expanding the use of confidential datasets across various domains. Their results demonstrated the effectiveness of the proposed algorithms in data anonymisation; however, the emphasis was on maintaining data structure and similarity for subsequent use in machine learning tasks. By contrast, the approach proposed in this study focuses on adapting the BlockSize, PrimeBase, and Mod parameters to ensure high accuracy, sensitivity, and efficiency in processing large-scale open data. Unlike M. Eleks *et al*., whose research assessed algorithms in terms of similarity preservation within a confidentiality framework, the adaptation of fuzzy hash function parameters in this study achieves high Precision, Recall, and F1 Score, which is crucial for detecting anomalies and duplicate records on government websites. Additionally, authors focused on developing algorithms for creating anonymised datasets that could be used to train machine learning models. In contrast, the present study is concerned with ensuring compliance with legal requirements, which necessitates not only preserving data structure but also achieving maximum accuracy in data analysis. In this context, fuzzy hash function parameter adaptation significantly reduces data processing time, distinguishing it from the work of M. Eleks *et al.* (2022), where processing speed was not the primary evaluation criterion.

K.V. Kumar *et al*. (2022) proposed an anonymous human activity recognition method that integrates deep neural networks with fuzzy hash algorithms. Their primary focus was on achieving high accuracy in real-time human action recognition, while employing the Recursive Genetic Micro-Aggregation Approach (RGMAA) model to enhance privacy protection. The Hybrid Deep Fuzzy Hashing Algorithm (HDFHA) used in their study enabled the detection of dependencies between different actions, improving overall accuracy. While their approach demonstrated high accuracy in processing dynamic video data, it requires significant computational resources due to the incorporation of the Deep Belief Network (DBN) and RGMAA. Although their methodology is effective for video data analysis, its computational complexity makes it less suitable for real-time text data monitoring. Both approaches leverage fuzzy hash functions but address different challenges. K.V. Kumar *et al*. focused on action recognition while ensuring data confidentiality, whereas the present study optimises fuzzy hash function parameters for text data analysis. The proposed optimisation demonstrated key advantages in terms of speed and accuracy, making it well-suited for monitoring open data on government websites.

T.-Z. Li *et al.* (2019) proposed a fuzzy hash function algorithm with adaptive file fragmentation, adjusting fragment size based on file dimensions. This enhancement improved the accuracy and efficiency of analysing both small and large files, making the approach particularly valuable in computer forensics, where precise similarity detection between files is crucial. The research presented in this paper, however, focused on evaluating the impact of modifying the BlockSize, PrimeBase, and Mod parameters on the efficiency of the fuzzy hash function, aiming to determine optimal values for texts of varying lengths. This approach considers the specific characteristics of open data on government websites, where text volume and structure can vary significantly. Identifying optimal parameters for different text lengths helps reduce collisions, increase accuracy, and enhance the sensitivity of the algorithm. In contrast to T.-Z. Li *et al.,* who optimised fragmentation rules for different file types, this study focused on large-scale text data, requiring a flexible approach to fuzzy hash function parameter configuration. Despite their differing focuses, both approaches contribute to the advancement of fuzzy

hashing methods, demonstrating effectiveness in their respective applications.

J. Chen *et al*. (2014) introduced a method for clustering spam campaigns using fuzzy hash functions. Their study aimed to detect spam botnets by grouping emails with similar content within a single spam campaign. The use of fuzzy hash functions enabled the successful processing of spam messages, even in the presence of obfuscation techniques such as URL shortening. Their research demonstrated the effectiveness of the proposed method on a three-year dataset comprising 540,000 spam emails, revealing typical behavioural patterns of botnets. In contrast, the approach proposed in this paper focuses on text data analysis, where adapting the fuzzy hash function parameters enhances analytical efficiency. While J. Chen *et al*. developed a powerful tool for spam campaign detection, the parameter adaptation in this study is critical for monitoring large-scale open data, where accuracy and processing speed are key factors.

Thus, the results of this study confirm that optimising the BlockSize, PrimeBase, and Mod parameters is an effective approach for enhancing accuracy, sensitivity, and processing speed in open data monitoring. The proposed solution offers a significant advantage over other approaches that rely on complex models or focus on different types of data. The implementation of adapted parameters enables superior performance, particularly in the context of government web resources, where ensuring high processing speed alongside high accuracy is essential.

Conclusions

This article examined methods for analysing the parameters of a fuzzy hash function to enhance the efficiency of monitoring compliance with the Regulation on Open Data on official government websites. The primary objective of the study was to determine the optimal values for BlockSize, PrimeBase, and Mod to ensure high accuracy and sensitivity in text processing, thereby improving open data monitoring. The study's findings were integrated into a software tool, leading to a significant improvement in accuracy and a reduction in data processing time. This was achieved through the dynamic adjustment of the block size and prime number base, depending on the length of the processed message.

The study also analysed how different values of BlockSize, PrimeBase, and Mod affected key algorithm performance metrics: Precision, Recall, F1 Score, and Processing Time. As a result, optimal parameter values were identified for different message lengths. For example, at a Message Length of 1,270, Precision was 0.85, increasing to 0.96 for a Message Length of 9,920, demonstrating enhanced accuracy in detecting similar data. The implementation of these optimised parameters in the software tool significantly improved the accuracy of detecting incorrect or duplicate data, which is crucial for automating compliance monitoring in accordance with open data legislation. Overall, the study demonstrated that fuzzy hash function parameter optimisation enhances data management quality and increases the efficiency of automated monitoring systems. The use of optimal parameters reduces the likelihood of errors in duplicate data detection while ensuring faster data processing.

Future research will focus on refining algorithms to achieve more precise parameter selection, tailored to the specific characteristics of the processed data. A particularly promising avenue is the adaptation of the algorithm for detecting duplicate media files (audio, video, and images) with appropriate pre-processing techniques. Advancements in this area have the potential not only to improve monitoring accuracy and completeness but also to reduce time and resource costs in data compliance verification. The development of enhanced methods for adapting fuzzy hash function parameters could significantly streamline data management and improve data quality, contributing to the more effective and reliable implementation of open data regulations.

Acknowledgements

None.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] AlMajali, A., Elmosalamy, A., Safwat, O., & Abouelela, H. (2024). Adaptive ransomware detection using similarity-preserving hashing. *Applied Sciences*, 14(20), article number 9548. doi: 10.3390/app14209548.

[2] Baba, T., Baba, K., & Yamauchi, T. (2022). Malware classification by deep learning using characteristics of hash functions. In: L. Barolli, F. Hussain & T. Enokido, (Eds.), *Advanced information networking and applications* (Vol. 450, pp. 480-491). Cham: Springer. doi: 10.1007/978-3-030-99587-4_40.

[3] Chanajitt, R., Pfahringer, B., Gomes, H.M., & Yogarajan, V. (2022). Multiclass malware classification using either static opcodes or dynamic API calls. In: H. Aziz, D. Corrêa & T. French (Eds.), *AI 2022: Advances in artificial intelligence* (Vol. 13728, pp 427-441). Springer, Cham. doi: 10.1007/978-3-031-22695-3_30.

[4] Chen, J., Fontugne, R., Kato, A., & Fukuda, K. (2014). Clustering spam campaigns with fuzzy hashing. In *Proceedings of the 10th Asian internet engineering conference* (pp. 66-73). New York: ACM. doi: 10.1145/2684793.2684803.

[5] Davies, S.R., Macfarlane, R., & Buchanan, W.J. (2021). Review of current ransomware detection techniques. In *Proceeding of the 7 th international conference on engineering and emerging technologies (ICEET)* (pp. 696-701). Istanbul: IEEE. doi: 10.1109/ICEET53442.2021.9659643.
[6] Eleks, M., Rebstadt, J., Fukas, P., & Thomas, O. (2022). Learning without looking: Similarity preserving hashing and its potential for machine learning in privacy critical domains. In *INFORMATIK 2022*, *lecture notes in informatics (LNI)* (pp.161-177). Bonn: IBiS. doi: 10.18420/inf2022_16.
[7] Fleming, M., & Olukoya, O. (2024). A temporal analysis and evaluation of fuzzy hashing algorithms for Android malware analysis. *Forensic Science International: Digital Investigation*, 49, article number 301770. doi: 10.1016/j.fsidi.2024.301770.
[8] Guerrero, M. (2022). Comparative study between Type-1 and interval Type-2 fuzzy systems in parameter adaptation for the Cuckoo search algorithm. *Symmetry*, 14(11), article number 2289. doi: 10.3390/sym14112289.
[9] Kida, M., & Olukoya, O. (2023). Nation-state threat actor attribution using fuzzy hashing. *IEEE Access*, 11, 1148-1165. doi: 10.1109/ACCESS.2022.3233403.
[10] Kondratenko, N.R. (2023). Interval type-2 generalizing fuzzy model for monitoring the states of complex systems using expert knowledge. *System Research and Information Technologies*, 2. doi: 10.20535/SRIT.2308-8893.2023.2.05.
[11] Kondratenko, N.R., & Snihur O.O. (2019). Research on the adequacy of interval type-2 fuzzy models in identifying complex objects. *System Research and Information Technologies*, 4, 94-104.
[12] Kumar, K.V., Harikiran, J., & Chandana, B.S. (2022). Human activity recognition with privacy preserving using deep learning algorithms. In *2nd international conference on artificial intelligence and signal processing (AISP)* (pp. 1-8). Vijayawada: IEEE. doi: 10.1109/AISP53593.2022.9760596.
[13] Li, T.-Z., Shen, B., Mi, K., Kao, Y.-C., & Cui, Y. (2019). A method of piecewise hash for fuzzy hashing. *Journal of Computers*, 30(2), 150-157. doi:10.3966/199115992019043002013.
[14] Mahrous, W.A., Farouk, M., & Darwish, S.M. (2021). An enhanced blockchain-based IoT digital forensics architecture using fuzzy hash. *IEEE Access*, 9, 151327-151336. doi: 10.1109/ACCESS.2021.3126715.
[15] Martín-Pérez, M., Rodríguez, R.J., & Breitinger, F. (2021). Bringing order to approximate matching: Classification and attacks on similarity digest algorithms. *Forensic Science International: Digital Investigation*, 36, article number 301120. doi: 10.1016/j.fsidi.2021.301120.
[16] Ministry of Digital Transformation of Ukraine. (n.d.). Retrieved from https://thedigital.gov.ua/.
[17] Naik, N., Jenkins, P., & Savage, N. (2019b). A ransomware detection method using fuzzy hashing for mitigating the risk of occlusion of information systems. *IEEE international symposium on systems engineering. (ISSE)* (pp. 1-6). Edinburgh: IEEE. doi:10.1109/ISSE46696.2019.8984540.
[18] Naik, N., Jenkins, P., Gillett, J., Mouratidis, H., Naik, K., & Song, J. (2019a). Lockout-Tagout Ransomware: A detection method for Ransomware using fuzzy hashing and clustering. *IEEE symposium series on computational intelligence (SSCI)* (pp. 641-648). Xiamen: IEEE. doi: 10.1109/SSCI44817.2019.9003148.
[19] Namanya, A.P, Awan, I.U., Disso, J.P., & Younas, M. (2020). Similarity hash based scoring of portable executable files for efficient malware detection in IoT. *Future Generation Computer Systems*, 110, 824-832. doi: 10.1016/j.future.2019.04.044.
[20] Nandal, A., Blagojevic, M., Milosevic, D., Dhaka, A., & Mishra, L.N. (2021). Fuzzy enhancement and deep hash layer based neural network to detect Covid-19. *Journal of Intelligent & Fuzzy Systems*, 41(1), pp. 1341-1351. doi: 10.3233/JIFS-210222.
[21] Natella, R. (2022). StateAFL: Greybox fuzzing for stateful network servers. *Empirical Software Engineering*, 27, article number 191. doi: 10.1007/s10664-022-10233-3.
[22] National Bank of Ukraine. (n.d.). Retrieved from https://bank.gov.ua.
[23] Open Data Portal. (n.d.). Retrieved from https://data.gov.ua.
[24] Pension Fund of Ukraine. (n.d.). Retrieved from https://www.pfu.gov.ua.
[25] Resolution of the Cabinet of Ministers of Ukraine No. 835 “On Approval of the Regulation on Data Sets Subject to Disclosure in the Form of Open Data”. (2015, October). Retrieved from https://zakon.rada.gov.ua/laws/show/835-2015-%D0%BF#Text.
[26] Ssdeep-project. (n.d.). *Fuzzy hashing API*. Retrieved from https://github.com/ssdeep-project/ssdeep.
[27] State Service of Special Communications and Information Protection of Ukraine. (n.d.). Retrieved from https://cip.gov.ua.
[28] State Statistics Service of Ukraine. (n.d.). Retrieved from https://ukrstat.gov.ua.
[29] State Tax Service of Ukraine. (n.d.). Retrieved from https://tax.gov.ua.
[30] Verkhovna Rada of Ukraine. Official Web Portal of the Parliament of Ukraine. (n.d.). Retrieved from https://www.rada.gov.ua/.

Дослідження параметрів нечіткої геш-функції для моніторингу дотримання вимог положення щодо відкритих даних

Леонід Майданевич

Кандидат філософських наук, старший викладач
Вінницький національний технічний університет
21021, вул. Хмельницьке шосе, 95, м. Вінниця, Україна
https://orcid.org/0000-0002-7364-8874

Наталія Кондратенко

Кандидат технічних наук, професор
Вінницький національний технічний університет
21021, вул. Хмельницьке шосе, 95, м. Вінниця, Україна
https://orcid.org/0000-0002-4450-1603

Віталій Казміревський

Аспірант
Вінницький національний технічний університет
21021, вул. Хмельницьке шосе, 95, м. Вінниця, Україна
https://orcid.org/0009-0005-4056-5385

Анотація. Метою роботи було дослідження параметрів геш-функції для підвищення ефективності та точності виявлення подібності текстових фрагментів на різних веб-ресурсах при проведенні моніторингу дотримання вимог Положення щодо відкритих даних на офіційних веб-сайтах державних органів. Дослідження охопило оцінку трьох ключових параметрів геш-функції: розміру блоку, бази простого числа та модуля. Для цього було проведено серію експериментів, у яких різні комбінації цих параметрів використовувалися для генерування геш-значень текстових даних. Результати дослідження продемонстрували, які комбінації параметрів забезпечують найкращий баланс між точністю, повнотою, F-мірою та часом виконання. Показано, що певні комбінації параметрів дозволяють досягти значного підвищення точності алгоритму при мінімізації обчислювальних витрат, що є важливим для аналізу даних у реальному часі. Встановлено, що оптимізація параметрів геш-функції сприяє зниженню кількості хибнопозитивних та хибнонегативних результатів, які часто виникають при виявленні подібності. Зокрема, підбір оптимальних значень для кожного з параметрів суттєво підвищує точність і повноту аналізу, дозволяючи отримати більш точні результати порівняння текстових фрагментів та зменшуючи час виконання операцій. Це робить алгоритм нечіткого гешування придатним для застосування в автоматизованих системах моніторингу державних веб-сайтів щодо дотримання вимог щодо відкритих даних. Виявлено, що оптимізація параметрів дозволяє зменшити кількість дубльованих записів, що особливо актуально для забезпечення відповідності відкритих даних вимогам законодавства. Одержані висновки можуть бути використані для розробки програмних засобів, які допоможуть ефективно виявляти недоліки та сприятимуть підвищенню прозорості та відповідності правовим вимогам. Крім того, результати дослідження можуть бути використані для подальшої оптимізації алгоритмів нечіткої геш-функції, що сприятиме вдосконаленню технологій моніторингу даних на відповідність нормативним вимогам. Дослідження робить внесок у розвиток технологій моніторингу веб-ресурсів, демонструючи, як правильно підібрані параметри нечіткої геш-функції можуть значно підвищити ефективність і надійність аналізу відкритих даних

Ключові слова: параметри нечіткої геш-функції; моніторинг веб-сайтів; державні електронні ресурси; точність алгоритму; параметри оптимізації; виявлення подібності; порушення положень