DOI: 10.31319/2519-8106.2(53)2025.342946
UDC 004.043

Yalova Kateryna, Candidate of Technical Sciences, Associate Professor, Head of the Department of the Systems software
Ялова К.М., кандидат технічних наук, доцент, кафедра програмного забезпечення систем
ORCID: 0000-0002-2687-5863
e-mail: yalovakateryna@gmail.com

Ismailov Vitaliy, PhD student, Department of the Systems software
Ісмаілов В.В., здобувач третього (доктора філософії) рівня вищої освіти, кафедра програмного забезпечення систем
email: ivv15081996@gmail.com

Sheliuh Kostiantyn, PhD student, Department of the Systems software
Шелюг К.Ю., здобувач третього (доктора філософії) рівня вищої освіти, кафедра програмного забезпечення систем
email: kostia902@ukr.net

Dniprovsky State Technical University, Kamianske
Дніпровський державний технічний університет, м. Кам'янське

ANALYSIS OF UNSTRUCTURED ADDRESS DATA USING FUZZY LOGIC ALGORITHMS

АНАЛІЗ НЕСТРУКТУРОВАНИХ АДРЕСНИХ ДАНИХ ЗАСОБАМИ АЛГОРИТМІВ НЕЧІТКОЇ ЛОГІКИ

The article is dedicated to the analysis of the effectiveness of applying the Damerau-Levenshtein and Jaro-Winkler algorithms for the processing of unstructured address data stored in a relational database. The study presents the algorithms for applying Damerau-Levenshtein and Jaro-Winkler to address matching and normalization tasks. For the numerical experiment, a reference dictionary of standard values was created, and a sample of 1000 records containing unstructured address data was used. To enable automatic address mapping, threshold values were set at 4 for Damerau-Levenshtein and 0.88 for Jaro-Winkler, after which the corresponding logical matching rules were formulated. The experimental results showed that the average address matching accuracy reached 0.9 for Jaro-Winkler and 0.72 for Damerau-Levenshtein. These findings indicate that Jaro-Winkler is more suitable for the preliminary stage of address normalization, especially in cases where addresses share a similar structure or common prefixes, since this algorithm assigns higher similarity to shared beginnings of strings. Therefore, Jaro-Winkler is recommended for shorter or structurally similar address names. At the same time, the Damerau-Levenshtein algorithm demonstrated lower average accuracy due to its higher sensitivity to character-level differences, yet it proved to be more effective for longer strings or cases involving transpositions and more complex variations in spelling.

***Keywords**: unstructured data, address data, fuzzy logic, Damerau-Levenshtein, Jaro-Winkler.*

У статті розглянуто проблему впорядкування та нормалізації адресних даних, що зберігаються у реляційних базах даних у неструктурованому вигляді. Така проблема є типовою для багатьох організацій, які працюють із великими обсягами інформації: різні формати введення, скорочення, помилки друку та перестановки елементів в адресних даних ускладнюють процеси пошуку й аналітики. З метою усунення зазначених недоліків запропоновано використання алгоритмів нечіткого порівняння рядків, здатних враховувати синтаксичні відмінності, зокрема, Damerau-Levenshtein та Jaro-Winkler. Метою дослідження є оцінювання ефективності застосування зазначених алгоритмів для нормалізації неструктурованих адресних даних, які зберігаються у реляційній базі даних. Основними завданнями дослідження є: розроблення алгоритмів застосування обраних метрик для аналізу адресних даних, проведення чисельного

експерименту, оцінювання точності та повноти співставлення, а також визначення переваг і обмежень кожного алгоритму. Основна ідея алгоритмів Damerau-Levenshtein та Jaro-Winkler полягає у визначенні схожості між вхідними та еталонними символьними рядками. У статті запропоновано алгоритми їхнього застосування для аналізу адресних даних, що зберігаються у реляційній базі даних, з урахуванням попередньої пренормалізації — приведення рядків до єдиного регістру, очищення від зайвих символів і стандартизації скорочень.

Для проведення чисельного експерименту був сформовано словник еталонних значень та використано вибірку з 1000 записів неструктурованих адресних даних. Для автоматичного мапування адрес були встановлено порогові значення 4 для Damerau-Levenshtein та 0,88 для Jaro-Winkler, після чого були сформовано відповідні логічні правила зіставлення. Для зручності графічного порівняння значення Damerau-Levenshtein було нормалізовано до діапазону [0,1]. Результати експерименту показали, що середня точність співставлення адрес становить 0,9 для Jaro-Winkler та 0,72 для Damerau-Levenshtein. Експеримент показав, що Jaro-Winkler частіше дає високі значення схожості, оскільки краще враховує збіг початкових частин слів (спільні префікси) і є ефективним при роботі зі скороченнями. Натомість Damerau-Levenshtein точніше відображає локальні орфографічні помилки на рівні символів. Отримані результати свідчать, що Jaro-Winkler є більш придатним для попереднього етапу нормалізації адрес, особливо у випадках, коли адреси мають подібну структуру або спільні початки, оскільки цей алгоритм враховує спільні префікси. Таким чином Jaro-Winkler доцільно застосовувати для коротких або схожих назв. Водночас алгоритм Damerau-Levenshtein продемонстрував нижчу середню точність через вищу чутливість до символьних змін, проте є більш ефективним для довших рядків або випадків, що містять перестановки та складніші відмінності у написанні.

***Ключові слова**: неструктуровані дані, адресні дані, нечітка логіка, алгоритм Damerau-Levenshtein, алгоритм Jaro-Winkler.*

Problem's formulation

The existence of the problem of unification and normalization of values in databases is a typical situation for many organizations that work with large volumes of information accumulated over an extended period of time. This issue becomes particularly critical in information systems that lack a unified data storage format and rely on manual data entry without validation or constraints, which leads to ambiguities and complicates further data processing. When working with address data, each component of an address (region, district, locality, street, building number) may be represented in different ways or combined into a single field. Such inconsistency not only complicates data search and sorting, but also affects subsequent analysis, processing, and integration with other systems.

Conventional string matching algorithms return a numerical similarity score; however, when a decision must be made regarding whether an input address is identical or not, it is advisable to apply fuzzy logic, which more closely resembles human reasoning processes. The analysis of address data under conditions of uncertainty requires the use of algorithms and approaches capable of accounting for syntactic differences, typographical errors, word rearrangements, grammatical variations, and word inflections, while focusing on small textual fragments. When processing unstructured address data, difficulties often arise due to the presence of alternative representations of the same address value. The most common challenges in the effective analysis of address data include inconsistent formatting, typographical errors, and the use of different abbreviations. An effective tool for addressing such issues is the use of fuzzy string matching or approximate string matching algorithms, which make it possible to compare textual values based on their degree of similarity rather than exact correspondence [1].

The search for effective approaches to identifying variations, alternative representations, duplicates, and input errors remains relevant in both theoretical and practical aspects.

Analysis of recent research and publications

In contemporary research, the analysis of unstructured address data is performed using fuzzy logic and edit distance algorithms to process erroneous, incomplete, or variable records. A separate group of scientific works consists of review studies that analyze theoretical foundations and applied solutions. In particular, R. Saatchi [2] and K. N. Kulkarni and R. K. Lad [3] described the concepts of fuzzy logic, its development, and practical implementations, emphasizing the importance of constructing interpretable rules and combining different metrics for working with textual data. Similarly,

Y.–W. Lai and M.–Y. Chen [4] analyzed the application of the fuzzy approach in text mining, identifying methods for data preprocessing and feature representation that are useful for subsequent address matching. In study [5], a framework was proposed for performing extract–transform–project operations on unstructured input data, enabling the application of structured reasoning to unstructured datasets.

Another group of studies focuses on specific algorithmic solutions and similarity metrics. M. Goswami and B. S. Purkayastha [1] presented a fuzzy model for the empirical analysis of unstructured data, demonstrating the construction of fuzzy sets and rules for classification. Among the works oriented toward concrete algorithms that investigated the Levenshtein and Jaro–Winkler distances are those by B. M. Addokali and E. A. Elburase [6], K. D Po [7], T. Petty [8], and O. Rozinek and J. Mares [9], which also presented methods for modifying these algorithms to improve accuracy. K. B. K Malaga [10] introduced an enhanced Jaro–Winkler algorithm with adapted variables for a domain-specific dictionary, demonstrating its efficiency in processing address elements. In study [11], the authors proposed applying text compression to signatures to reduce the computational complexity of using the Levenshtein distance while maintaining the accuracy of the comparison.

Separately, a number of works have examined practical examples of address matching and normalization. K. Ramani and D. Borrajo [12] described a complete workflow for processing English–language addresses, including preprocessing, tokenization, and combined candidate ranking based on fuzzy rules. K. Lee [13] presented an approach that integrates machine learning with address geocoding, improving the accuracy and robustness of results. Additionally, A. Makalesi [14] analyzed the combination of N–gram techniques and fuzzy aggregation for extracting information from documents such as scanned invoices or forms. R. P. Kandregula [15] explored various string-distance metrics and indexing systems, emphasizing that hybrid approaches provide the best results for address matching.

Thus, modern approaches to processing unstructured address data can be categorized into three main groups: review studies analyzing conceptual frameworks, research focused on similarity algorithms aimed at improving matching accuracy, and applied studies describing the integration of fuzzy logic methods with machine learning and indexing for scalable solutions. Among the key trends are the combination of character–based and token–based metrics with fuzzy aggregation, and the implementation of optimization techniques for processing large volumes of address data.

Formulation of the study purpose

The purpose of the paper is to present the results of evaluating the effectiveness of applying fuzzy logic methods, specifically the Damerau–Levenshtein distance (DLD) and Jaro–Winkler distance (JWD), for the normalization of unstructured address data stored in a relational database. The objectives of the study are as follows: to develop algorithms for applying DLD and JWD to analyze address data stored in a database, to conduct a numerical experiment, and to evaluate the obtained results.

Presenting main materials

The task of analyzing unstructured address data belongs to the class of fuzzy string matching tasks, for which the following groups of algorithms are typically applied:

1. Edit distance–based algorithms, which calculate the minimum number of operations required to transform one string into another. Examples include the Levenshtein distance and DLD.

2. Character–based comparison algorithms, which evaluate the similarity of strings based on the number of common characters or matching sequences, such as the Jaro distance and JWD.

Traditional comparison methods tend to emphasize differences between strings, whereas fuzzy algorithms are capable of identifying similarity between records even in the presence of typographical or structural variations. Among the most effective algorithms in this context are DLD and JWD. DLD measures the minimum number of edit operations required to transform an input string $s=s_1,..,s_m$ into another input string $t=t_1,..,t_n$. It accounts for four types of operations: insertion, deletion, substitution, and transposition (the exchange of adjacent symbols) [6]. The definition of the DLD is given as follows:

$$d(i,j)=\min\begin{cases} d(i-1,j)+1 \\ d(i,j-1)+1 \\ d(i-1,j-1)+cost \\ d(i-2,j-2)+1 \end{cases}, \quad (1)$$

where $d(i,j)$ — is the minimum cost of transforming the first i characters of string s into the first j characters of string t; *cost* represents the conditional cost of a character substitution operation, which is used to determine whether substitution should be performed during matching [11].

The task of applying DLD for analyzing address data stored in a relational database is reduced to computing the minimal number of insertion, deletion, substitution, or transposition operations required to transform an input address into its reference form. The developed algorithm for the normalization of unstructured data using DLD can be described by the following main steps:

1. Reading data from the database table.
2. Data pre-normalization:
   a. converting address characters to lowercase;
   b. removing delimiters and extra spaces;
   c. unifying abbreviations of address components.
3. Computing the DLD according to formula (1):
   a. for each pair of addresses — input s and reference t — the DLD value is calculated.
4. Selecting the most probable variant:
   a. for each input address, the candidate with the minimum DLD value is selected;
   b. if DLD ≤ the predefined threshold, the correspondence is considered valid.
5. Writing results into the table of normalized address data.

The output data represent the distance between strings, expressed as the value of the minimum number of edit operations. The advantage of the DLD lies in its ability to accurately account for local spelling errors and character transpositions, which makes it particularly useful in cases where data distortion occurs at the level of individual letters. However, DLD is insensitive to the grammatical order of words and morphological variations, which reduces its effectiveness when comparing grammatically correct but syntactically different forms, for example: «Lesi Ukrainky» and «Ukrainky Lesi».

The most appropriate and effective application areas of DLD include:

- normalization of address names;
- automatic correction of errors already contained in address databases;
- fuzzy search in postal and geoinformation systems.

The JWD algorithm is used to determine the degree of similarity between two strings, for example, when comparing approximately entered unstructured textual data such as addresses. It takes into account the number of common characters, their order, and the length of the common prefix between two input strings $s=s_1,..,s_m$ and $t=t_1,..,t_n$ [9]. This algorithm is particularly effective when processing addresses containing abbreviations such as «pr-t Lesi Ukrainky» and «prospekt Lesi Ukrainky». The output of the algorithm is a similarity coefficient — a decimal number ranging from 0 to 1, indicating the degree to which s is similar to t [10]. Mathematically, JWD is defined as:

$$JW = J + \left(l \cdot p \cdot \left(1 - J\right)\right), \tag{3}$$

where l — is the length of the common prefix at the beginning of the string, the maximum value of which is up to 4 characters; that is, if the string begins identically, the output value increases, p — is the weight of the prefix, J — is the Jaro metric, which is defined as:

$$J = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), \tag{4}$$

where s_1, s_2 — are the strings of values, $|s_1|, |s_2|$ — the lengths of the respective strings, m — is the number of matching characters, and t is the number of transpositions when characters are present in the strings but arranged in a different order. In the context of the task of normalizing unstructured address data stored in a relational database, the application of JWD reduces to measuring the similarity of address fields based on the number of shared characters and the distance between them. The algorithm for applying JWD to the analysis of address data is described by the following steps:

1. Reading data from the database table;
2. Data pre–normalization:
   a. Converting address characters to lowercase;
   b. Removing delimiters and extra spaces;

c. Unifying abbreviations of address components;

3. Calculation of JWD and application of the Winkler adjustment to account for prefixes according to formulas 3 and 4;

4. Searching for the nearest reference address:

a. For each input address, find the reference address with the highest JWD;

b. If JWD ≥ established threshold, consider the match correct;

5. Recording the results in the table of normalized address data.

A weakness of JWD is its low resistance to word rearrangement and strong dependence on the order of characters. However, in tasks where the priority is the rapid selection of potentially similar strings, the algorithm proves to be quite productive and computationally efficient [15]. Based on the calculated coefficient, the following operations can be performed:

– searching for duplicates in the address database;
– linking records across different datasets;
– automatic correction and generation of suggestions for address entry.

The purpose of the numerical experiment was to convert unstructured address data, recorded with errors, in various formats, with omissions and duplications, stored in a relational database, into a single reference format. The input data for the experiment was the supplier table in the database with the field address_supplier, which was populated via manual data entry without using input masks or validation. The table contained 1,000 records. The address_supplier field data were not normalized. The output data was the normalized_address table, in which each address from the supplier table was associated with the most similar reference address using DLD and JWD. The constructed reference address dictionary was represented as the reference table with a single field address. For automatic mapping of address data, thresholds were set as DLD = 4 and JWD = 0,88. The condition for automatic mapping was:

– if DLD ≤ 4 or JWD ≥ 0,88, perform automatic mapping and save the results in the normalized_address table;
– otherwise, mark the address field value for manual review.

Tabl. 1 presents a brief fragment of the experimental results, showing examples of input and reference address pairs, as well as the calculated values of the DLD and JWD metrics.

Table 1. Fragment of experimental results

№	Input address from supplier.address_supplier	Reference address from reference.address	DLD	JWD
1	Vul. Shevchenka 12 Kyiv	vulytsia shevchenka, 12, kyiv	8	0.952
2	prosp. Peremohy 46 Kharkiv	prospekt peremohy, 45, kharkiv	5	0.928
3	Vulytsia Svobody 3 Lvov	vulytsia svobody, 3, lviv	2	0.96
4	vul. Mirna 9 Odessa	vulytsia myrna, 10, odesa	6	0.89
5	Pr. Nezalezhnosty 25 Kiev	prospekt nezalezhnosti, 25, kyiv	9	0.92
6	Vul. Soborna 8 Vinitsia	vulytsia soborna, 8, vinnytsia	6	0.90
7	vul. Khreshchatyk 1 Kiiv	vulytsia khreshchatyk, 1, kyiv	7	0.94
8	Vulytsia Ukrainska 15 Zaporizhzhia	vulytsia ukrainska, 14, zaporizhzhia	2	0.98
9	Vul. Franka 6 Poltava	vulytsia franka, 6, poltava	4	0.95
10	Vulytsia Kobzaria 4 Ivano-Frankivsk	vulytsia kobzaria, 4, ivano-frankivsk	0	1.00

For the visual representation of data and accurate comparison, the DLD values were normalized so that both coefficients fall within the same range of values [0,1]. The normalized value DLD* was calculated using the following formula:

$$DLD^{*} = 1 - \frac{DLD(s,t)}{\max(len(s), len(t))}, \tag{4}$$

where *len*(*s*), *len*(*t*) — denote the respective lengths of the string values.

Fig. 1 presents the results of calculating the similarity coefficients for 1000 pairs of address records, where the *x*–axis and *y*–axis represent the index of the address pair, and the similarity coefficient.

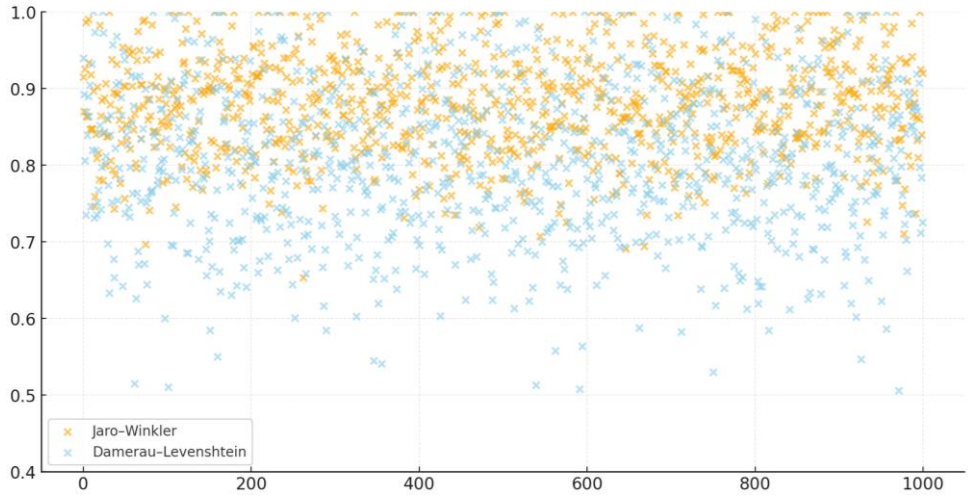

Fig. 1. Comparison of similarity metric calculation results

Based on the results of the experimental analysis, it can be concluded that the JWD metric demonstrates higher average values and a smaller variance, indicating that most of the stored addresses shared similar beginnings or structural patterns. In contrast, the DLD metric exhibits a wider range of values with greater variability, as the DLD value decreases even with minor insertions, deletions, or character transpositions, i.e., it is more sensitive to structural discrepancies in address data.

The precision values for JWD are higher than those for DLD, suggesting that this algorithm effectively identifies similar pairs. However, due to its sensitivity to initial matches in strings, the recall values are lower than those of DLD, especially in cases where pairs contain character transpositions or differ in length. The precision values for DLD were mostly within the range of 0.6—0.85, which is lower than those of JWD, since the algorithm produces more false positives for pairs that appear similar but are not semantically identical. At the same time, recall values are higher for DLD due to its ability to account for edit distances and to detect similarity even when strings differ.

Conclusions

Fuzzy matching algorithms are powerful tools for analyzing textual data under conditions of uncertainty. They significantly improve the quality of address record comparison by mitigating issues of duplication, ambiguity, and input errors. Combining several algorithms enables achieving higher accuracy in complex tasks of address data normalization. The experimental results indicate that JWD is better suited for the preliminary stage of address normalization, particularly when the addresses share similar structures or common prefixes, as JWD prioritizes shared beginnings. Therefore, JWD is more appropriate for short or structurally similar names. DLD, on the other hand, showed lower values due to its higher sensitivity but performs better for longer strings or cases involving value transpositions.

Based on the analyzed data, conceptual provisions were formulated regarding the feasibility of developing a hybrid approach to applying fuzzy logic algorithms, since both metrics complement each other. Their combination can enhance the effectiveness of unstructured address data normalization by efficiently handling various types of errors and variations in the input address data.

References

[1] Goswami, M., & Purkayastha, B. S. (2020). A fuzzy based approach for empirical analysis of unstructured data. Journal of computational and theoretical nanoscience, 17(9), 4375—4379. doi: 10.1166/jctn.2020.9080.

[2] Saatchi, R. (2024). Fuzzy logic concepts, developments and implementation. Information, 15(10), 1—24. doi: https://doi.org/10.3390/info15100656.

[3] Kulkarni, K. N, & Lad, R. K. (2021). Fuzzy logic and its developmental advances: a review. Proceedings of The 2nd International Conference on IoT Based Control Networks & Intelligent Systems (ICICNIS`21), Kerala, India.

[4] Lai, Y.–W., & Chen, M.–Y. (2023). Review of survey research in fuzzy approach for text mining. IEEE Access, 11, 39635—39649. doi: 10.1109/ACCESS.2023.3268165.

[5] Sadia, M., Chowdhury, A. R., & Chen, A. (2025). A case for computing on unstructured data. arXiv, 2509, 1—6.

[6] Addokali, B. M., & Elburase, E. A. (2022). Using Levenshtein Distance Algorithm to increase database search efficiency and accuracy. Research Gates, 10, 1—7.

[7] Po, K. D. (2020). Similarity based information retrieval using distance algorithm. International Journal of Advances in Scientific Research and Engineering, 6(4), 6—10.

[8] Petty, T., Hanning, J., Huszar, T., & Lyer, H. (2022). A new string edit distance and applications. Algorithms, 15(7), 1—22. doi: https://doi.org/10.3390/a15070242.

[9] Rozinek, O., & Mares, J. (2024). Fast and precise convolutional Jaro and Jaro-Winkler Similarity. Proceedings of The 35th Conference of Open Innovations Association (FRUCT`24), Tampere, Finland. doi: 10.23919/FRUCT61870.2024.10516360.

[10] Malaga, K. B. K., Verdillo, K. L., Pascual, E. S. (2025). An enhancement of the Jaro-Winkler fuzzy searching algorithm applied in library search engine. Journal of Information Systems Engineering and Mangement, 10(28), 649—660. doi: https://doi.org/10.52783/jisem.v10i28s.4369.

[11] Coates P., & Breitinger F. Identifying document similarity using a fast estimation of the Levenshtein Distance based on compression and signatures. Proceedings of the Digital Forensics Research Conference Europe (DFRWS EU`22), Oxford, UK.

[12] Ramani, K., & Borrajo, D. (2024). Methods for matching English language addresses. arXiv, 2403.12092, 1—15.

[13] Lee, K., Claridades, A. R. C., & Lee, J. (2020). Improving a street-based geocoding algorithm using machine learning techniques. Applied Science, 10(16), 5628. doi: https://doi.org/10.3390/app10165628.

[14] Makalesi, A. (2021). Comparison of different classification algorithms for extraction information from invoice images using an N-gram approach. European Journal of Science and Technology, 31(1), 991—1003. doi: 10.31590/ejosat.844862.

[15] Kandregula, R. P. (2021). Comparison of Apache SOLR search, spellcheck string distance measure — Levenshtein, Jaro–Winkler, and N–Gram. International Journal of Computer Trends and Technology, 69(3), 1—4. doi: https://doi.org/10.14445/22312803/ IJCTT-V69I3P101.

Список використаної літератури

1. Goswami M., Purkayastha B. S. A fuzzy based approach for empirical analysis of unstructured data. *Journal of computational and theoretical nanoscience*. 2020. №17(9). P. 4375—4379. doi: 10.1166/jctn.2020.9080.
2. Saatchi R. Fuzzy logic concepts, developments and implementation. *Information*. 2024. №15(10). 1—24. doi: https://doi.org/10.3390/info15100656
3. Kulkarni K. N, Lad R. K. Fuzzy logic and its developmental advances: a review. *Proceedings of the International Conference on IoT Based Control Networks & Intelligent Systems — ICICNIS 2021*. PP. 1—4
4. Lai Y–W., Chen M.–Y. Review of survey research in fuzzy approach for text mining. *IEEE Access*. 2023. №11. P. 39635—39649. doi: 10.1109/ACCESS.2023.3268165

5. Sadia M., Chowdhury A. R., Chen A. A case for computing on unstructured data. *arXiv*. 2025. № 2509. P. 1—6.
6. Addokali B. M., Elburase E. A. Using Levenshtein Distance Algorithm to increase database search efficiency and accuracy. *Research Gates*. 2022. №10. P. 1—7.
7. Po K. D. Similarity based information retrieval using distance algorithm. *International Journal of Advances in Scientific Research and Engineering*. 2020. 6(4). P. 6—10.
8. Petty T., Hanning J., Huszar T., Lyer H. A new string edit distance and applications. *Algorithms*. 2022. №15(7). P. 1—22.doi: https://doi.org/10.3390/a15070242
9. Rozinek O., Mares J. Fast and precise convolutional Jaro and Jaro–Winkler Similarity. 2024. 35th Conference of Open Innovations Association (FRUCT), Tampere, Finland. pp. 604—613, doi: 10.23919/FRUCT61870.2024.10516360.
10. Malaga K. B. K., Verdillo K. L., Pascual E. S. An enhancement of the Jaro–Winkler fuzzy searching algorithm applied in library search engine. *Journal of Information Systems Engineering and Mangement*. 2025. №10(28). P. 649—660. doi: https://doi.org/10.52783/jisem.v10i28s.4369
11. Coates P., Breitinger F. Identifying document similarity using a fast estimation of the Levenshtein Distance based on compression and signatures. Proceedings of the Digital Forensics Research Conference Europe (DFRWS EU), March 29—April 1, 2022. P. 1—11.
12. Ramani K., Borrajo D. Methods for matching English language addresses. *arXiv*. 2024. 2403.12092. P.1—15
13. Lee K., Claridades A.R.C., Lee J. Improving a street-based geocoding algorithm using machine learning techniques. *Applied Science*. 2020. 10(16). 5628; https://doi.org/10.3390/app10165628
14. Makalesi A. Comparisom of different classification algorithms for extraction information from invoice images using an N–gram approach. *European Journal of Science and Texhnology*. 2021. 31(1). P. 991—1003. doi: 10.31590/ejosat.844862
15. Kandregula R.P. Comparison of Apache SOLR search, spellcheck string distance measure — Levenshtein, Jaro–Winkler, and N–Gram. *International Journal of Computer Trends and Technology*. 2021. Vol. 69., no. 3. P. 1—4. doi: https://doi.org/10.14445/22312803/ IJCTT-V69I3P101

Надійшла до редколегії 15.10.2025
Прийнята після рецензування 21.10.2025
Опублікована 23.10.2025