УДК 004.02
DOI https://doi.org/10.32782/tnv-tech.2023.3.3

МОДИФІКАЦІЯ АЛГОРИТМІВ НЕЧІТКОГО ПОШУКУ ДЛЯ ВИКОРИСТАННЯ ТАБЛИЦІ ПОДІБНОСТІ СИМВОЛІВ

***Клещ К. О.** – асистент та аспірант кафедри системного проектування,*
інституту прикладного системного аналізу
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID ID: 0009-0006-8133-3086

***Царьов М. О.** – бакалавр кафедри системного проектування,*
інституту прикладного системного аналізу
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID ID: 0009-0007-1092-4791

Об'єктом дослідження є алгоритм нечіткого пошуку на основі відстані Дамерау-Левенштейна та таблиця подібності символів. У роботі було досліджено, проаналізовано та надано рекомендації, як саме можна інтегрувати потужності таблиці подібності символів з алгоритмом нечіткого пошуку Дамерау-Левенштейна. Дослідження алгоритмів нечіткого пошуку в тексті є важливою темою в галузі інформаційного пошуку та обробки тексту. Це обумовлено зростаючим обсягом інформації і ймовірністю помилок через вплив людського фактора під час написання тексту. Нечіткий пошук використовує алгоритми для пошуку даних в тексті, які приблизно відповідають шаблону. Це досягається шляхом порівняння та зіставлення рядків або ключових слів, які можуть бути схожими, але не ідентичними. Для нечіткого пошуку можна використати таблицю подібності символів, яка допомагає з'ясувати міру подібності пари символів. Поєднуючи алгоритм нечіткого пошуку з таблицею подібності, можна досягти більш точного та індивідуального доступу до великого обсягу інформації, яка зберігається у текстовому форматі.

В роботі було проведено порівняльний аналіз ефективності та коректності результатів алгоритмів нечіткого пошуку з використанням таблиці і без її використання, а також алгоритму точного пошуку. Використання таблиці подібності покращує отримані результати, особливо при використанні мов зі спеціальними символами. Це дозволяє знаходити значно більше релевантних результатів, проте швидкодія алгоритму зменшується. Отримані результати могли б стати важливим внеском у вдосконалення пошукових систем. Це дозволило б користувачам знаходити відповідні документи навіть при наявності орфографічних помилок, синонімів, скорочень або інших форм неточностей у запиті. Підхід з використанням таблиці подібності символів міг би бути використаний у системах для перевірки орфографії та автоматичної корекції, системах автопропозицій та автозавершення, а також у реалізації функцій з виявлення плагіату та дублікатів даних.

***Ключові слова:** нечіткий пошук, таблиця подібності символів, алгоритм Дамерау-Левенштейна, обробка текстових даних, відстань редагування.*

Kleshch K. O., Tsarov M. O. Modification of the fuzzy search algorithms to use a symbols similarity table

The object of the research is a fuzzy search algorithm based on the Damerau-Levenshtein distance and a symbols similarity table. The work involved investigating, analyzing, and providing recommendations on how to integrate the capabilities of the symbols similarity table with the Damerau-Levenshtein fuzzy search algorithm. Researching of the fuzzy search algorithms in text is an important topic in the field of an information retrieval and text processing. This is driven by the increasing volume of the information and the likelihood of errors due to the human factors during text composition. Fuzzy search utilizes algorithms to find data in the text that matches patterns approximately. This is achieved by comparing and matching strings or keywords that

may be similar but not identical. A symbols similarity table can be employed for fuzzy search that helps in determining the degree of similarity between pairs of characters. By combining the fuzzy search algorithm with the symbols similarity table, more accurate and personalized access to a large volume of the text-based information can be achieved.

The study conducted a comparative analysis of the effectiveness and correctness of the fuzzy search algorithms with and without the using of a symbols similarity table, as well as an exact search algorithm. The using of the symbols similarity table improves the obtained results, especially when there are languages featuring special characters. This allows to find finding significantly more relevant results, although the speed of the algorithm decreases. The obtained results could contribute significantly to the enhancement of the search systems. This could enable users to find relevant documents even in the presence of spelling errors, synonyms, abbreviations, or other forms of inaccuracies at the query. The approach of the utilizing a symbols similarity table could be used in systems for spelling checking and automatic correction, auto-suggestion and auto-completion systems, as well as into the implementing functions of the plagiarism detection and data duplicates.

Keywords: *fuzzy search, symbol similarity table, Damerau-Levenstein algorithm, text data processing, editing distance.*

Вступ. Алгоритми нечіткого пошуку потрібні для пошуку приблизних результатів для заданого шаблону в тексті, навіть якщо точної відповіді в наборі даних немає. Алгоритм Дамерау-Левенштейна – один з найбільш популярних алгоритмів нечіткого пошуку. Головна ідея полягає у визначенні відстані редагування між кожним словом в початковому тексті та заданим рядком, який називається пошуковим словом або зразком [1], використовуючи метрику Дамерау-Левенштейна. Відстань Дамерау-Левенштейна показує відмінність між двома послідовностями символів та обчислюється як мінімальна кількість операцій: вставок, видалень, замін та транспозицій (перестановок сусідніх символів), необхідних для перетворення однієї послідовності на іншу [2]. Відстань Дамерау-Левенштейна є модифікацією класичної відстані Левенштейна. До трьох стандартних операцій додана операція транспозиції. Дамерау обґрунтував свій підхід у роботі [3], де він стверджує, що понад 80% орфографічних помилок в інформаційно-пошукових системах були вчинені саме через один із чотирьох типів помилок. Для обчислення відстані Дамерау-Левенштейна часто застосовується алгоритм динамічного програмування. Для цього створюється і заповнюється матриця розміром (n + 1) * (m + 1), де n і m – довжини порівнюваних рядків. Вартість чотирьох наведених операцій вважається однаковою і дорівнює одиниці. Проте при реалізації алгоритму це можна змінити, що і буде реалізовано за допомогою таблиці подібності символів. Недоліком використання лише відстані Дамерау–Левенштейна для пошуку в тексті є те, що даний алгоритм не враховує семантику або контекст тексту, що може призводити до неточних результатів [4]. Наприклад, під час пошуку слова “tree” з відстанню редагування 1, може знайтись результат “free”, що не підходить за контекстом.

Алгоритм нечіткого пошуку з використанням таблиці подібності є модифікацією алгоритму Дамерау-Левенштейна, в якому для порівняння міри подібності символів використовується ця таблиця. Даний алгоритм враховує як і візуальну та семантичну подібність окремих символів, так і операції вставки, видалення, заміни і транспозиції для вимірювання загальної подібності між рядками.

В рамках дослідження розглядалась задача пошуку слів серед великого обсягу текстових даних з використанням таблиці подібності символів. Яка дозволить коректніше знаходити результати пошуку для семантично подібних символів або тих, що розташовані поряд на клавіатурі, за рахунок змешення ціни такої похибки. Головною метою роботи було створити та впровадити таблицю подібності

символів в алгоритм нечіткого пошуку на основі відстані Дамерау-Левенштейна. А також провести заміри швидкодії та повноти результатів для алгоритмів нечіткого пошуку з використанням такої таблиці та без неї. Це дасть змогу покращити використання алгоритмів нечіткого пошуку, особливо для мов зі спеціальними символами.

Реалізація таблиці подібності символів. Таблиця подібності символів, також відома як таблиця відображення символів, є довідковою таблицею, яка чисельно визначає подібність між символами. Завдяки ній можна визначити міру подібності символів з різних мов чи просто візуально подібних символів [5]. Мета таблиці подібності символів полягає в тому, що вона є важливим інструментом для транслітерації, транскрипції та конвертації між різними системами письма. Найбільш поширеною структурою таблиці подібності символів є матриця, де кожен рядок і стовпець відповідають символам, а в комірках матриці зберігається оцінка подібності між відповідними парами символів [6]. Оцінка подібності може вказувати на те, наскільки близькими є два символи за їхньою формою, звуком або іншими характеристиками. Значення в таблиці подібності варіюються від 0 до 1, де 1 означає відсутність подібності, а 0 – ідеальний збіг [7]. Така таблиця повинна бути симетричною, оскільки подібність між символом "a" і символом "b" така ж, як між символом "b" і символом "a". Формулу для модифікованого алгоритму Дамерау-Левенштейна можна побачити на рис. 1, де compareCharCost(a, b) – міра подібності між символами "a" та "b", яку можна отримати з таблиці подібності.

$$d_{a,b}(i,j) = min \begin{cases} 0 \;\; if\; i = j = 0, \\ d_{a,b}(i-1,j) + 1 \;\; if\; i > 0, \\ d_{a,b}(i,j-1) + 1 \;\; if\; j > 0, \\ d_{a,b}(i-1,j-1) + compareCharCost \;\; if\; i,j > 0, \\ d_{a,b}(i-2,j-2) + 1 \; if\; i,j > 1 \; and \; a_i = b_{j-1} \; and \; a_{i-1} = b_j \end{cases}$$

Рис. 1. Формула модифікованого алгоритму Дамерау-Левенштейна

Класична структура таблиці подібності на основі матриці не підійшла через погану швидкодію. Така таблиця була не оптимальною з точки зору обсягів пам'яті для її зберігання, а також асимптотичної складності перевірки подібності між двома символами [8]. Тому в рамках роботи була створена інша таблиця подібності, структура якої зображена на рис. 2. Дана таблиця складається з двох словників. В першому словнику ключем є символ, для якого потрібно знайти йому подібні символи, а значенням – другий словник. В другому словнику ключем є символи, подібні до ключа з першого словника, а значенням ціна їх подібності. Дана структура буде зберігати властивість симетричності, не дивлячись на те, що це призведе до зберігання дублікатів даних. Асимптотична складність знаходження символу у даній структурі буде амортизовано константною O(1) [9].

Для прикладу розглянемо деякі подібні символи до англійської літери "a":

1. "A" – та сама літера, але велика
2. "ä" – схожий символ, проте з іншої мови
3. "s" – літера поруч на клавіатурі

Реалізована таблиця подібності буде складатися з трьох різних категорій подібності символів.

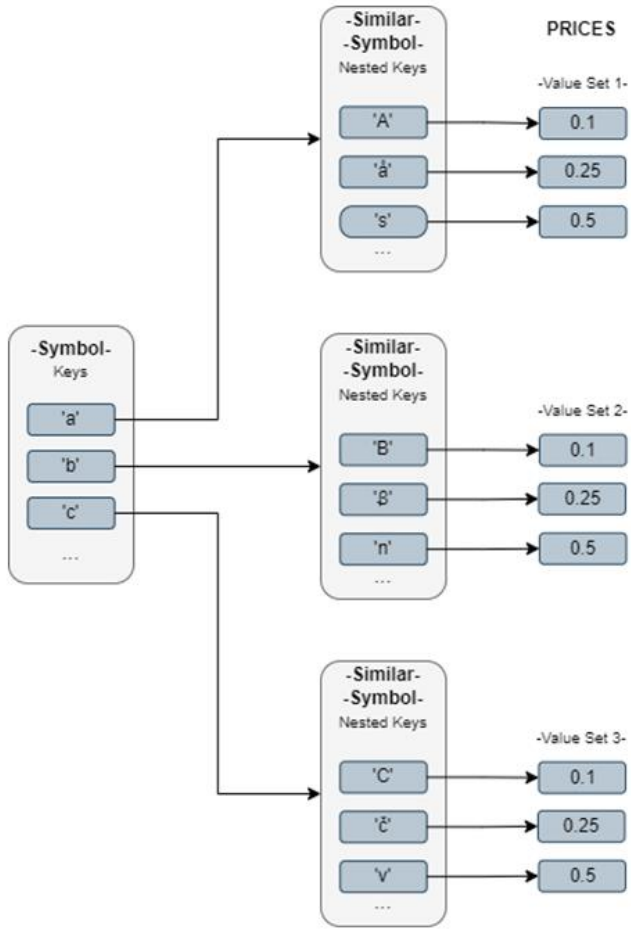

Рис. 2. Структура таблиці подібності символів

1) Семантично подібні символи

В дану категорію будуть входити семантично схожі символи з різних мов. За основу було взято джерело [10]. Оскільки це електронний ресурс, то для побудови таблиці подібності найлегше та найефективніше його проаналізувати буде за допомогою мови програмування Python. Саму таблицю подібності можна зберігати в форматі JSON, оскільки завдяки цьому можна легко серіалізувати дані таблиці, а потім легко десеріалізувати.

2) Символи, які поруч на клавіатурі

Дана категорія подібних символів буде містити можливі друкарські помилки для англійського алфавіту. Кожна група подібних символів в даній категорії буде складатися з символу та всіх можливих помилок, а саме : всіх оточуючих символів на клавіатурі.

3) Візуально подібні символи

Цю категорію символів потрібно заповнювати вручну, заздалегідь прописавши в таблицю всі візуально подібні символи. До візуально подібних можна,

наприклад, віднести наступну пару символів "1", "l" (одиниця та англійська маленька літера "л").

Результати та їх обговорення. Тестування алгоритму нечіткого пошуку з використанням таблиці подібності можна розділити на дві категорії: тестування таблиці подібності та тестування обчислення відстані редагування. Для проведення тестування розробленого алгоритму було обрано одну з найпопулярніших бібліотек GoogleTest. GoogleTest – фреймворк тестування для мови C++, який розроблено командою Testing Technology з урахуванням конкретних вимог і обмежень Google [11]. Розглянемо тестування правильності роботи таблиці подібності. Для цього заздалегідь пропишемо значення подібності для всіх пар символів, які тестуються. Після цього потрібно перевірити, чи повертає таблиця подібності очікувані значення для всіх пар символів. Для тестування обчислення відстані редагування також було розроблено тест, який перевіряє роботу алгоритму Дамерау-Левенштейна з таблицею подібності. В даному тесті перевіряється можливі випадки пар слів для обчислення їхньої відстані редагування. Для перевірки правильності роботи функції всі розрахунки відстані Дамерау-Левенштейна було проведено вручну та звірено з результатами функції. Для прикладу пара слів "anthropology" та "anthopolögy". В другому слові маємо дві помилки – перша пропуск символу "r", друга – заміна літери "o" на літеру з іншої мови. В результаті, ціна перетворення другого слова на перше дорівнює 1.25, що і повинна повернути функція.

В якості фінального тесту перевіримо кількість результатів які повертають різні алгоритми пошуку: Дамерау-Левенштейна з таблицею, Дамерау-Левенштейна без таблиці і звичайний чіткий алгоритм пошуку. Для цього підготуємо великий набір текстових даних перекладений на 3 різні мови, а саме: англійську, німецьку та чеську. І будемо дивитись на кількість результатів для різних пошукових слів з різними типами помилок. Результати такого тестування зображено для чеської мови на рис. 3., для німецької мови – на рис. 4., та для англійської мови – на рис. 5.

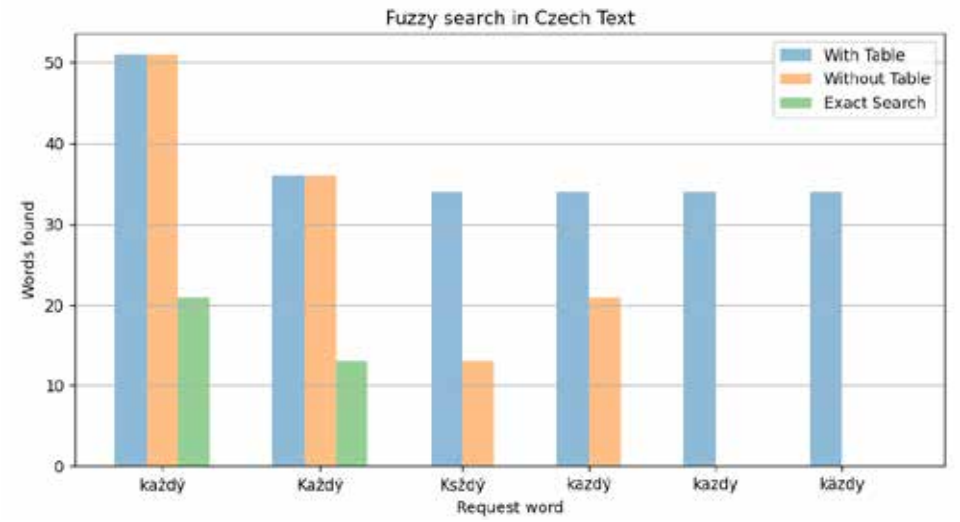

Рис. 3. Результати фінального тестування для чеської мови

По х-осі на даних гістограмах зображені різні пошукові запити, а по осі у – кількість знайдених слів. Дивлячись на дані гістограми, можна дійти висновку, що при слові, яке не має жодної модифікації, обидва алгоритми показують однаковий результат. Проте, коли ми змінюємо пошуковий запит, нечіткий пошук з використанням таблиці показує значно кращий результат, ніж алгоритм без таблиці. При цьому алгоритм чіткого пошуку знаходить на порядок менше слів, а при допущенні декількох помилок взагалі видає 0 релевантних результатів. Також можна сказати, що чим більше змінених символів у слові, тим меншу кількість слів можна знайти.

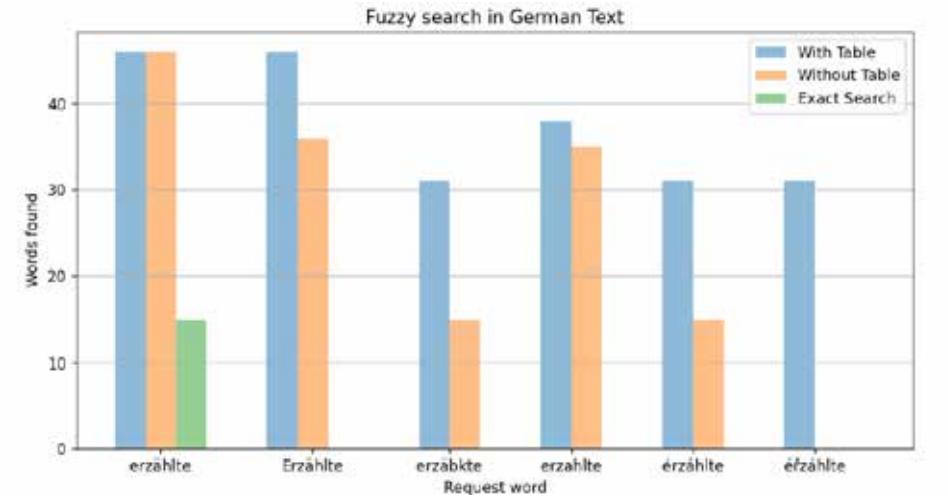

Рис. 4. Результати фінального тестування для німецької мови

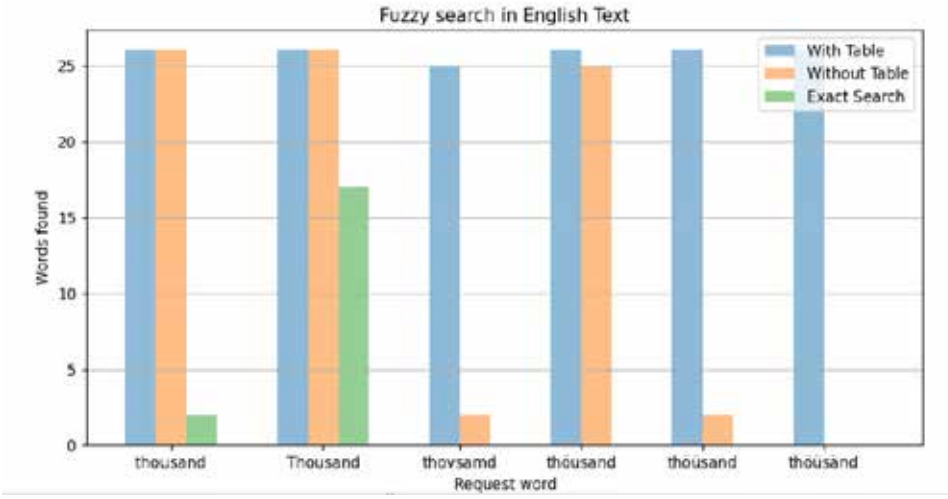

Рис. 5. Результати фінального тестування для англійської мови

Тест швидкодії показав, що використання таблиці подібності зменшує продуктивність алгоритму на 10-20% в залежності від довжини пошукового слова. Це спричинено тим, що потрібно робити додаткові запити в таблицю подібності для кожної пари символів. Проте значно більша кількість знайдених релевантних результатів компенсує цей недолік.

Висновки. Жоден з існуючих алгоритмів нечіткого пошуку не враховує можливу семантичну подібність символів в словах. Таблиця подібності символів виявилася ефективним рішенням для вирішення цієї проблеми, фіксуючи можливі семантичні та інші подібності між символами. Оцінюючи подібності на основі різних критеріїв, таких як: форма, контекст і фонетика, таблиця подібності символів дозволила модифікувати алгоритм нечіткого пошуку, який зміг би ідентифікувати та ранжувати відповідні результати навіть за наявності помилок у вхідному запиті чи тексті, в якому відбувається пошук.

В рамках дослідження було модифіковано алгоритм Дамерау-Левенштейна, додавши до розрахунку відстані редагування між словами, можливість отримати міру схожості символів з таблиці подібності. Для реалізації такого підходу було створено саму таблицю подібності та реалізовано модифікований алгоритм Дамерау-Левенштейна.

В якості тестування між собою порівнювались 3 різні алгоритми пошуку з точки зору швидкодії та кількості знайдених релевантних результатів. А саме: алгоритм Дамерау-Левенштейна з таблицею подібності, без таблиці та класичний алгоритм чіткого пошуку. Аналіз було здійснено на великому наборі тестових даних 3 різними мовами. Перший алгоритм показує значно кращі результати нечіткого пошуку за знаходженням релевантних результатів, проте другий виконує пошук на такому ж наборі даних в середньому на 10-20% швидше. При цьому алгоритм чіткого пошуку показав низькі результати при внесенні певних помилок в пошукові слова чи в текст.

Загалом, порівнюючи ці алгоритми, можна дійти висновку, що нечіткий пошук з використанням таблиці подібності слід використовувати, коли для користувача більш важливі результати пошуку. А пошук без таблиці слід використовувати, коли користувачу важлива саме швидкість обчислення. Водночас чіткий пошук найкраще підходить для задачі пошуку зразка в тексті при відсутності помилок різних типів.

В якості перспективи розвитку алгоритму нечіткого пошуку в тексті з використанням таблиці подібності можна створити функціонал індексування текстових даних, наприклад, використовуючи BK-дерева. Це дозволить збільшити швидкодію нечіткого пошуку з використанням таблиці.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. М. В. Михайлова. Порівняння алгоритмів нечіткого пошуку в текстах українською мовою: Радіоелектроніка, інформатика, управління, 2007, 80 с.

2. Відстань Дамерау-Левенштейна [Електронний ресурс]. https://www.geeksforgeeks.org/damerau-levenshtein-distance/

3. Fred J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors : Communications of the ACM, 1964, с. 171 – 176.

4. Gonzalo N. A guided tour to approximate string matching / Navarro Gonzalo. // Association for Computing Machinery. – 2001.

5. J. P. Carvalho and L. Coheur, "Introducing UWS – A fuzzy based word similarity function with good discrimination capability: Preliminary results," *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Hyderabad, India, 2013, pp. 1-8.

6. H. Ayeldeen, A. E. Hassanien and A. A. Fahmy, "Lexical similarity using fuzzy Euclidean distance," *2014 International Conference on Engineering and Technology (ICET)*, Cairo, Egypt, 2014, pp. 1-6.

7. Mihov S. Fast Approximate Search in Large Dictionaries / S. Mihov, K. Schulz. // Computational Linguistics. – 2004.

8. Yu, M., Li, G., Deng, D. et al. String similarity search and join: a survey. Front. Comput. Sci. 10, 399–417 (2016).

9. Вступ в алгоритми, 4 видання / [Т. Кормен, Р. Рівест]., 2022. – 1312 с.

10. Fancy Letters [Електронний ресурс]. https://symbl.cc/en/collections/fancy-letters/

11. Посібник користувача Google Benchmar [Електронний ресурс]. https://github.com/google/benchmark/blob/main/docs/user_guide.md#runtime-and-reporting-considerations

REFERENCES:

1. M. V. Mykhailova. Comparison of fuzzy search algorithms in Ukrainian texts: Radio electronics, computer science, management, 2007, 80 s.

2. Damerau–Levenshtein distance. URL: https://www.geeksforgeeks.org/damerau-levenshtein-distance/

3. Fred J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors : Communications of the ACM, 1964, c. 171 – 176.

4. Gonzalo N. A guided tour to approximate string matching / Navarro Gonzalo. // Association for Computing Machinery. – 2001.

5. J. P. Carvalho and L. Coheur, "Introducing UWS – A fuzzy based word similarity function with good discrimination capability: Preliminary results," *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Hyderabad, India, 2013, pp. 1-8.

6. H. Ayeldeen, A. E. Hassanien and A. A. Fahmy, "Lexical similarity using fuzzy Euclidean distance," *2014 International Conference on Engineering and Technology (ICET)*, Cairo, Egypt, 2014, pp. 1-6.

7. Mihov S. Fast Approximate Search in Large Dictionaries / S. Mihov, K. Schulz. // Computational Linguistics. – 2004.

8. Yu, M., Li, G., Deng, D. et al. String similarity search and join: a survey. Front. Comput. Sci. 10, 399–417 (2016).

9. Introduction to Algorithms, fourth edition / [T. Cormen, C. Leiserson, R. Rivest]., 2022. – 1312 c.

10. Fancy Letters. URL: https://symbl.cc/en/collections/fancy-letters/

11. Google Benchmar User Guide. URL: https://github.com/google/benchmark/blob/main/docs/user_guide.md#runtime-and-reporting-considerations