ISSN 2664-9969

UDC 004.02
DOI: 10.15587/2706-5448.2024.298425

Kyrylo Kleshch

DEVELOPMENT OF FUZZY SEARCH METHOD FOR CREATING AN EFFICIENT INFORMATION SEARCH SYSTEM IN TEXT DATA

The object of research is the processes of effective search for information in a set of textual data. The subject of the research is the fuzzy search method, which will allow to effectively solve the problem of searching for information in a set of textual data. The paper considers the process of developing a fuzzy search method, which consists of 9 consecutive steps and is required for a quick search for matches in a large set of text data. Based on this method, it is proposed to create a fuzzy search system that will solve the problem of finding the most relevant documents from a set of such documents.

The proposed fuzzy search method combines the advantages of algorithms based on deterministic finite automata and algorithms based on dynamic programming for calculating the Damerau-Levenshtein distance. Such a combination allows to implement the symbol similarity table in an optimal way. As part of the work, an approach for creating a symbol similarity table was proposed and an example of such a table was created for symbols from the English alphabet, which allows to find the degree of similarity between two symbols with constant asymptotics and to convert the current symbol into its basic counterpart. For document filtering, a metric was developed to evaluate the correspondence of text data to a search phrase, which simultaneously takes into account the number of found and not found characters and the number of found and not found words.

The Damerau-Levenstein algorithm allows to find the edit distance between two words, taking into account the following types of errors: substitution, addition, deletion, and transposition of characters. The work proposed a modification of this algorithm by using a similarity table to more accurately estimate the editing distance between two words.

The developed method makes it possible to create a fuzzy search system that will help find the desired results faster and increase the relevance of the obtained results by sorting them according to the values of the proposed test data similarity metric.

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

Received date: 16.12.2023
Accepted date: 11.02.2024
Published date: 13.02.2024

How to cite
Kleshch, K. (2024). Development of fuzzy search method for creating an efficient information search system in text data. Technology Audit and Production Reserves, 1 (2 (75)), 20–24. doi: https://doi.org/10.15587/2706-5448.2024.298425

1. Introduction

The volume of electronic information in the world grows annually by 30 %, accordingly, the increase in the amount of available data complicates the process of managing and searching for information in this data [1]. Since access to the necessary information is of great importance, the skills of efficient search in large volumes of data become extremely important. When working with textual data such as documents, web pages, e-mails, tables, presentations, etc., it is critical to be able to quickly and accurately find the information and documents necessary from a list.

However, traditional search algorithms can be ineffective, especially on large datasets, or when there are inaccuracies in the search phrase or textual data, such as spelling or typographical errors. In such cases, classical algorithms that rely on exact matches may not meet the needs of users [2]. That is why algorithms for fuzzy text search are becoming extremely important. The study of fuzzy text search algorithms is an important direction in the field of information search and text processing [3]. Fuzzy search technology is used to search information and text databases, providing matches for a pattern, even in the presence of errors or uncertainties in the data [2].

There are many fuzzy search algorithms such as:
– Jaro-Winkler algorithm;
– N-gram algorithm;
– Bitap algorithm;
– usual Levenshtein algorithm.

However, they were not suitable for the implementation of the proposed method for various reasons. The fuzzy search method, which combines the advantages of algorithms based on deterministic finite automata and algorithms based on dynamic programming for calculating the Damerau-Levenshtein distance, performed best. Such a combination allows to implement the symbol similarity table in an optimal way.

As part of the study, the task of finding the most relevant text documents and e-mails in the environment for storing such documents was considered. The user must be able to find information about the necessary document without knowing the correct spelling of a particular word, and not paying attention to possible inaccuracies in the textual data.

The aim of research is to develop a fuzzy search method that will allow to perform the task in an optimal way, as well as to implement the proposed method in the system for searching for the most relevant e-mails, documents or texts in the backup environment. To implement such a method, it is necessary to modify the Damerau-Levenshtein algorithm by introducing a symbol similarity table.

2. Materials and Methods

2.1. Problem statement. *The object of research* is the processes of effective search for information in a set of textual data. The main task of the work is the development of a fuzzy search method that will help find the desired results faster and increase the relevance of the obtained results, due to their sorting according to the values of the test data similarity metric.

The task was considered in the following form. Let X be the set of documents that are in the data storage environment D, which is presented in Fig. 1. Documents can be of different types: doc, email, pptx, txt, pdf, html, image, cpp, etc., the main thing is that each document contains text data in its title or content. S is a user-supplied search phrase, also called a template. A pattern can consist of a single word, several words that form a sentence, or a set of independent words. The search phrase can be specified with any unicode character.

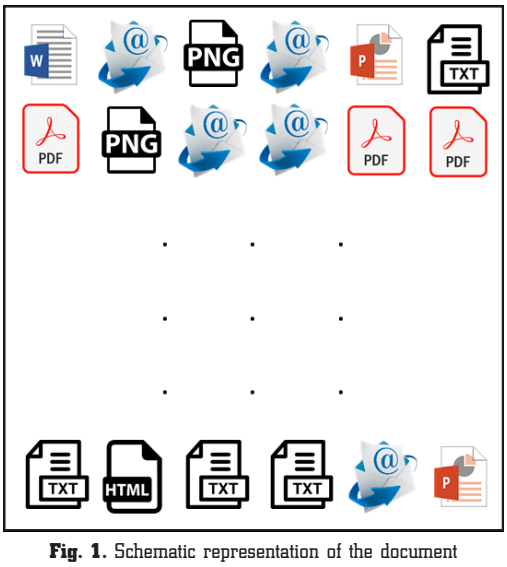

Fig. 1. Schematic representation of the document storage environment

Then $Y \in X$ is a subset of documents from X that are most relevant to the search phrase S and are sorted in order of decreasing relevance according to the textual data similarity metric.

2.2. Conducting an experiment. In practice, a fuzzy search method was implemented, which consists of 9 steps and combines the advantages of various algorithms, both based on deterministic finite automata and dynamic programming algorithms for calculating the Damerau-Levenshtein distance. These algorithms were implemented in the C++ programming language and implemented in the document search system in the data backup environment. The proposed method was tested for execution time and correct operation using an expert system.

2.3. Symbol similarity table. A symbol similarity table is a reference table that numerically determines the similarity between symbols. It allows to identify similar symbols from different languages or simply visually similar symbols. Usually, the values in the similarity table are defined from 0 to 1, where zero indicates a perfect match, and one indicates no similarity [4]. The paper proposes the approach of filling the table in the form of groups in a JSON file. To improve the efficiency of work, it is necessary to create a table of similarity of symbols, which will consist of several different categories:

– *Semantically similar symbols.* The category will consist of semantically similar symbols from different languages to symbols from the English alphabet, an example of symbols is shown in Fig. 2.
– *Typographical errors.* The category will contain possible typographical errors for characters from the English alphabet, namely all surrounding characters on the keyboard, an example of the characters is shown in Fig. 3.
– *Visually similar symbols.* This category consists of groups of symbols that have a similar appearance, an example is shown in Fig. 4.

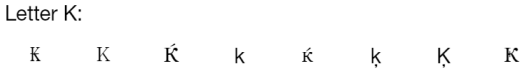

Fig. 2. An example of semantically similar symbols for the letter «Ķ» [5]

Fig. 3. An example of typographical errors

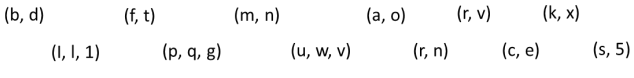

Fig. 4. An example of visually similar symbols

The symbol similarity table consists of two dictionaries. Such a structure allows to perform operations with constant asymptotics. The table has 2 main functions: the first accepts two characters as input, and returns a measure of their similarity from 0 to 1; the second accepts a character, and returns the base for it. For example, let the user set the search phrase *S=«Každý»* (everyone), then its base counterpart will be *S(base)=«kazdy»*. «For such a transformation, it is necessary to translate each character from the search phrase into its basic counterpart using the similarity table function.

2.4. Fuzzy search based on deterministic finite automata. At this stage, it is necessary to choose a fuzzy search algorithm that will quickly find all words from the text set

ISSN 2664-9969

that are close to the search word. That is, with the editing distance <=*d_max*. Algorithms based on finite automata are ideal for the current task. After all, such algorithms build all possible patterns of words that differ from the search word by no more than a given editing distance.

In practice, 3 different solutions based on finite state machines were tested. A tree-based automaton implements the idea of constructing a finite automaton based on a prefix tree. It accepts all words that differ from the search phrase by no more than a given edit distance. At the same time, taking into account the operations of insertion, deletion, transposition and replacement. The main disadvantage of this machine is a large number of states for a long search word and an editing distance of more than 3 [6]. Accordingly, it takes a long time to build and consumes a lot of memory. Due to the listed shortcomings, it was decided to abandon the use of this machine in the proposed method.

The second approach implements an automaton based on hashing and, accordingly, there is no binding to the structure of the prefix tree. Let the cost of each operation, be it deletion, insertion, transposition or replacement, be the same and equal to one, which allows to build a more structured automaton, helping to speed up the construction and determinization of NFA [7]. Each state of the automaton corresponds to a certain configuration, which includes the number of processed characters in the pattern and the number of editing operations used. Each transition between states corresponds to a specific operation. States that have fully processed the pattern are final. The proposed automaton is quite similar in principle to the Levenstein automaton, but at the output, in addition to the Boolean value, it also returns the editable distance between words, and also supports the symbol transposition operation. Among the disadvantages, one can note the complex selection of a hash function for hashing states and a long determinization process [7]. However, these shortcomings are not critical, and this approach is proposed as a priority for solving the given problem.

The logic of building an automaton based on a transition table is no different from the logic of building an automaton based on hashing. The main advantage of using a transition table is the absence of the need for explicit determinism [7]. With a template and a maximum editing distance, the machine can immediately start checking words. When analyzing each symbol of the word, a characteristic vector is calculated relative to the given pattern. Depending on the value of this vector and the current state, the next state of the automaton is determined, which is selected from the table of all possible transitions [7]. After all incoming characters have been processed, a check is made to see if the current state is final. This approach is only practical for small values of the maximum edit distance, since the size of the table grows exponentially.

3. Results and Discussion

To implement the proposed method, it is necessary to perform the following steps:

1. Get a set of text data from incoming documents. Split the text into words.
2. Convert each symbol from the set to its basic counterpart using a similarity table.
3. Translate the search phrase into a basic analogue using the similarity table.
4. Build a deterministic finite automaton for each word from the search phrase.
5. Use DFA to find the previous edit distance between each word in the search phrase and the text set.
6. Use the modified Damerau-Levenshtein algorithm to specify the editing distance.
7. Match words from the search phrase with words from the text set.
8. Calculate the numerical metric of relevance of text data to the search phrase.
9. Sort and filter incoming documents according to the obtained characteristics.

The first step is to get the text from the incoming documents. If the input document is already a text file or a web page, then no pre-processing of the data is assumed. If the incoming documents are pictures or presentations, then the file name can be taken as text data. After that, it is necessary to divide the received text into words, using separator symbols. The second and third steps of the proposed method are performed using a similarity table. In the fourth step, it is necessary to build a deterministic finite automaton based on hashing for each search word.

After passing the fifth step of the proposed method, the vast majority of words from the text set will be filtered out, because they will have an edit distance greater than the maximum allowable suitable edit distance. However, there will still be words that could potentially fit, and it is for such words that the edit distance needs to be refined using a modified Damerau-Levenshtein algorithm using a similarity table.

A modification of the Damerau-Levenshtein algorithm is that a symbol similarity table is also used to calculate the editing distance. This allows for more detailed control over character similarity comparisons, allowing the algorithm to handle cases where certain characters are visually or semantically similar to others [8]. In the case of replacing a symbol, the degree of similarity between two symbols is determined using a table, this is reflected in the formula using the compareCharCost() function, which is shown in Fig. 5.

Let's consider an example: let the search phrase be the word *«Každý»*, and the word from the text data set be *«kazdy»*, then it is necessary to find the edit distance between these words using the classic Damerau-Levenshtein algorithm and the modified one, the results of the work are shown in Fig. 6 and Fig. 7, respectively.

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

Fig. 5. Formula for calculating the modified Damerau-Levenshtein distance

		K	a	ž	d	ý
	0	1	2	3	4	5
k	1	1	2	3	4	5
a	2	2	1	2	3	4
z	3	3	2	3	3	4
d	4	4	3	3	2	3
y	5	5	4	4	3	3

Fig. 6. Calculation of the edit distance using the classic Damerau-Levenshtein algorithm

		K	a	ž	d	ý
	0	1	2	3	4	5
k	1	0.25	1.25	2.25	3.25	4.25
a	2	1.25	0.25	1.25	2.25	3.25
z	3	2.25	1.25	0.45	1.45	2.45
d	4	3.25	2.25	1.45	0.45	1.45
y	5	4.25	3.25	2.45	1.45	0.65

Fig. 7. Calculation of the editing distance by the modified Damerau-Levenshtein algorithm

After that, it is necessary to match each word from the template with each word from the text, in Fig. 8 words from the template are marked in blue, and words from the text are marked in red.

Let's introduce the word similarity metric:

$$P_{word}(a, b) = 1 - d / max(len(a), len(b)),$$

where d – the modified Damerau-Levenshtein distance; $len(x)$ – the length of word x.

For the same words, the metric will show 1, for completely different words – 0. For each word from the search phrase, let's calculate the metric with each word from the text.

Possible situations:

1) there is no match for the current word in the text, so the current word is not found;

2) there is an unambiguous match in the text for the current word, accordingly, the current word is found, the word in the text is marked as used;

3) for the current word there are several matches in the text, accordingly the current word is found, the word in the text with a higher metric is marked as used.

Let's introduce the phrase similarity metric. It is worth noting: the situation when there is no match for the search word from the search phrase in the text is much worse than the situation when not all words from the text are marked as used [9]:

$$P_{phrase} = \frac{\left(\sum_{Sword} P_{word} \cdot len(word)\right)}{len(phrase)} - \frac{\left(\sum_{Tword} (1 - P_{word}) \cdot len(word)\right)}{len(text) \cdot tFactor} \cdot penaltyCoef,$$

where $tFactor$ and $penaltyCoef$ are balancing coefficients.

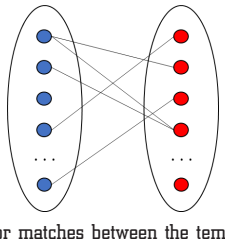

Fig. 8. Search for matches between the template and the text

The metric can take values between 0 and 1, where 1 is the best match and 0 is the worst. In the case of too large a penalty, the metric may even become negative, in which case it is possible to consider it equal to zero.

Let's introduce the relevance metric of phrases $R_{phrase} = (1 - P_{phrase}) \cdot maxScore$, the metric can take on values from 0 – the best match, to $maxScore$ – the worst match, the value of $maxScore = 100'000$ is proposed in the work.

To improve the accuracy of the calculation, it is suggested to introduce a module that will ignore certain words in a long search phrase and in a long text. For example, words from the English language that do not make sense when searching: «a», «the», «in», «at», «by», «to», but the following short words cannot be ignored: «my», «you», «no», «yes».

Based on the proposed metrics, it is necessary to calculate the relevance of each document to the search phrase, and then sort and filter only those whose metric value is less than the threshold value. The method of expert evaluation was used to select the balancing coefficients. Several expert users were offered an input test, namely: a search phrase and a set of textual data. Their task was to arrange the set of input data in the order of decreasing relevance to the search phrase, discarding, in their opinion, irrelevant instances [10]. Based on the opinion of experts, the order of phrases for each test, which is considered true, was formed. The system was balanced so that all test cases returned results in this order. For example, consider the search phrase «His eyes functioned fine» and a set of input text data:

- «human eye»;
- «cognitive functions»;
- «his book»;
- «fine-tuning»;
- «the eyes»;
- «fine movements»;
- «eye functions»;
- «absent or non-functioning»;
- «eye to see fine detail»;
- «his spiritual eyes».

The result of the system operation for such inputs is shown in Fig. 9, respectively, the most relevant results with more matches are located above. It should be noted that the system filters results that are not at all relevant to the search phrase.

```
Phrase: His eyes functioned fine
eye functions - 51869
cognitive functions - 73905
absent or non-functioning - 77368
eye to see fine detail - 78854
his spiritual eyes - 79420
the eyes - 89524
fine-tuning - 92952
fine movements - 94799
his book - 99048
```

Fig. 9. System performance result

The proposed system was successfully implemented in the C++ programming language and implemented in a web application for storing backup copies of data. Thanks to the system, the average search time for relevant results has decreased by 5 %, but for search phrases of different lengths and different number of words in the phrase, the result may differ. The longer the search phrase, the smaller the gain in time. The number of relevant results increased by 10 %, but this estimate is highly dependent on the input text data and the frequency of unforced spelling errors by the user.

ISSN 2664-9969

The limitations of the study include the small number of experts who participated in the formation of the system for determining the correct order of sorting documents according to the search phrase, based on this system, the values of the coefficients in the proposed formulas were determined.

The conditions of the martial law in Ukraine in no way affected the obtained results, because the proposed method is based on known algorithms, which were made certain modifications. However, the war in Ukraine has affected the search queries that users make more often and the information stored in documents and web pages in cloud environments.

Other modifications of fuzzy search algorithms are planned in the future. For example, it is possible to add the use of b-trees or the use of technologies for working with big data. Separately, it is worth considering the direction of context analysis in documents, because fuzzy search algorithms are trained only to search for words with possible errors, but they do not know how to work with synonyms.

4. Conclusions

A fuzzy search method is proposed for finding the most relevant documents according to a search phrase. The main idea of the method is that first it is necessary to translate the search phrase and the input text sets into their basic counterparts using a table of character similarities. Using DFA, determine those words from the text that have an edit distance to the search words less than a given value. Only for these words from the text, determine the refined editing distance using the modified Damerau-Levenshtein algorithm.

To implement several steps of the method, an approach to creating a symbol similarity table was proposed, which would allow taking into account the possible semantic similarity of symbols in words. By evaluating similarities based on different criteria such as form, context, and phonetics, the character similarity table allowed a modified fuzzy search algorithm that can rank matching results even in the presence of a large number of unicode mismatches of similar character values. This, in turn, increased the number of relevant results by 10 %, depending on the length of the search phrase.

To select the most suitable documents, a metric for evaluating the relevance of text data according to the search phrase was proposed. On the basis of testing with the help of expert evaluation, the optimal coefficients in the relevant formulas were calculated.

Conflict of interest

The author declares that he has no conflict of interest in relation to this research, including financial, personal, authorship, or any other nature that could affect the research and its results presented in this article.

Financing

The study was conducted without financial support.

Data availability

The manuscript has no associated data.

Use of artificial intelligence

The author confirms that he did not use artificial intelligence technologies when creating the presented work.

References

1. Boytsov, L. (2011). Indexing methods for approximate dictionary searching. *ACM Journal of Experimental Algorithmics, 16*. doi: https://doi.org/10.1145/1963190.1963191
2. Carvalho, J. P., Coheur, L. (2013). Introducing UWS – A fuzzy based word similarity function with good discrimination capability: Preliminary results. *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Hyderabad. doi: https://doi.org/10.1109/fuzz-ieee.2013.6622494
3. Yu, M., Li, G., Deng, D., Feng, J. (2015). String similarity search and join: a survey. *Frontiers of Computer Science, 10 (3)*, 399–417. doi: https://doi.org/10.1007/s11704-015-5900-5
4. Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys, 33 (1)*, 31–88. doi: https://doi.org/10.1145/375360.375365
5. *Fancy Letters*. Available at: https://symbl.cc/en/collections/fancy-letters/
6. Snášel, V., Keprt, A., Abraham, A., Hassanien, A. E. (2009). Approximate String Matching by Fuzzy Automata. *Advances in Soft Computing*. Berlin Heidelberg: Springer, 281–290. doi: https://doi.org/10.1007/978-3-642-00563-3_29
7. Kleshch, K., Shablii, V. (2023). Comparison of fuzzy search algorithms based on Damerau-Levenshtein automata on large data. *Technology Audit and Production Reserves, 4 (2 (72))*, 27–32. doi: https://doi.org/10.15587/2706-5448.2023.286382
8. Kleshch, K. O., Tsarov, M. O. (2023). Modification of the fuzzy search algorithms to use a symbols similarity table. *Taurida Scientific Herald*. Series: Technical Sciences, 3, 21–28. doi: https://doi.org/10.32782/tnv-tech.2023.3.3
9. Mihov, S., Schulz, K. U. (2004). Fast Approximate Search in Large Dictionaries. *Computational Linguistics, 30 (4)*, 451–477. doi: https://doi.org/10.1162/0891201042544938
10. Wang, J., Li, G., Fe, J. (2011). Fast-join: An efficient method for fuzzy token matching based string similarity join. *2011 IEEE 27th International Conference on Data Engineering*. Hannover, 458–469. doi: https://doi.org/10.1109/icde.2011.5767865

Kyrylo Kleshch, *Assistant, Postgraduate Student, Department of System Design, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, ORCID: https://orcid.org/0009-0006-8133-3086, e-mail: kleshch.kirill@gmail.com*