Matrix: Jurnal Manajemen Teknologi dan Informatika
Volume 11 Issue 2 Year 2021 Pages 63-70
DOI: http://dx.doi.org/10.31940/matrix.v11i2.2419
URL: http://ojs.pnb.ac.id/index.php/matrix

Designing a word recommendation application using the Levenshtein Distance algorithm

Nadhia Nurin Syarafina [1]*, Jozua Ferjanus Palandi [2], Nira Radita [3]

[1,2,3] *Teknik Informatika STIKI Malang, Indonesia*

*Corresponding Author: *nurinnadhia@gmail.com*

Abstract: Good scriptwriting or reporting requires a high level of accuracy. The basic problem is that the level of accuracy of the authors is not the same. The low level of accuracy allows for mistyping of words in a sentence. Typing errors caused the word to become non-standard. Even worse, the word became meaningless. In this case, the recommendation application serves to provide word-writing recommendations in case of a typing error. This application can reduce the error rate of the writer when typing. One method to improve word spelling is Approximate String Matching. This method applies an approach to the string search process. The Levenshtein Distance algorithm is a part of the Approximate String-Matching method. This method, firstly, is necessary to go through the preprocessing stage to correct an incorrectly written word using the Levenshtein Distance algorithm. The application testing phase uses ten texts composed of 100 words, ten texts composed of 100 to 250 words, and ten texts composed of 250 to 500 words. The average accuracy rate of these test results was 95%, 94%, and 90%.

Keywords: levenshtein distance, approximate string matching, preprocessing, recommendation, "PUEBI"

History Article: Submitted 9 March 2021 | Revised 1 May 2021 | Accepted 21 May 2021

How to Cite: N.N.Syarafina, J.F. Palandi, and N. Radita, "Designing a word recommendation application using the Levenshtein Distance algorithm," *Matrix: Jurnal Manajemen Teknologi dan Informatika*, vol. 11, no. 1, pp. 64–71, 2021.

Introduction

Typing a report manuscript or an essay requires high accuracy. The level of accuracy of a writer has different levels. A low level of accuracy can result in errors in typing a term or word, which makes a term or a phrase non-standard, and may even have no meaning. Also, the factor that causes typing errors is a habit of a writer in abbreviating a word. This typing error problem often occurs when a writer writes scientific papers, proposals, or reports for school, college, or work needs. This kind of error is fatal if not corrected immediately. The wrong word can have a different meaning, or it can even be a word with a different connotation in a sentence. Spelling errors and writing non-standard terms can change the true meaning of information and lead to readers' misinterpretation [1]. We can avoid these typing errors by using an Approximate String Matching method, which is a method for matching a string based on similarity in terms of writing, both the number of characters and the composition of characters in a document [2]. This method has several types of algorithms, one of which is the Levenshtein Distance algorithm. The Levenshtein Distance algorithm has three types of string operations, namely deleting, adding, and changing. These operations are useful for calculating the distance between 2 strings. The smaller the distance between the two strings indicates that the two strings are match [3]. Fixing this problem can use the Levenshtein Distance algorithm with a high level of accuracy in terms of similarity search words. An application to recommend the right words in a text using the Levenshtein Distance algorithm is the right solution to solve this problem. This application helps minimize word writing errors, both words that are not standard or words that have no meaning.

The Indonesian language has developed very rapidly as a result of advances in science, technology, and art. The use of the Indonesian language is increasingly widespread, both verbally and in writing. Therefore, writing a manuscript requires a reference or a guideline in writing an Indonesian

language script. In this case, Language Development and Development Board has made improvements to the Indonesian spelling. The refinement resulted in a document called "Pedoman Umum Ejaan Bahasa Indonesia (PUEBI)" [4].

A spelling checker is a feature that checks words or spelling errors based on a specific language. The spelling checker looks for all kinds of mistakes in the document, then warns the user regarding the existing error and provides some suggestions to fix it [5]. Furthermore, spell checker features include a) proofreading and parsing of all words in the manuscript, b) comparing parsing results with a dictionary containing a list of correct word spelling, c) handling errors by utilizing appropriate matching techniques [6].

In the field of image processing, preprocessing means the initial process for removing noise [7]. However, the preprocessing function in this program is to get keywords that serve to match a string or to compare a document [8]. In general, the preprocessing stage consists of four steps: case folding, tokenizing, filtering, and stemming [9]. In this study, the filtering stage was eliminated so that the preprocessing process only consisted of tokenizing, case folding, and stemming. Tokenizing is generally replacing sensitive data with a unique identification symbol and stores all important information about the data [10]. Based on NLP (Natural Language Processing) research, tokenization is a technique for dividing documents into separate tokens, either with spaces or punctuation, which are useful in identifying meaningful keywords and using them for further processing [11]. The omitted input strings are numbers, symbols, and punctuation. Case folding is a stage that converts text from uppercase to lowercase [12]. Stemming is an advanced level used to remove prefixes or suffixes [10] into root words [12].

Approximate String Matching is a method in computer science applied in text search, pattern recognition, and signal processing applications [13] that allow the search results to match the given keywords [14]. Even though the results are not the same, Approximate String Matching successfully finds mismatches in typing or recommendations if the exact word was not found [14]. The approximate string matching approach is a generalization of the exact string matching approach [15]. The reasons for introducing approximate string matching are: low quality of the text, heterogeneousness of databases, spelling errors in the pattern or text, searching for foreign names, and searching with uncertainty [15].

Levenshtein distance algorithm was invented by Vladimir Levenshtein, a scientist from Russia, in 1965 [16]. The edit distance calculation is obtained from a matrix that calculates the number of differences between two strings. This algorithm runs from the top left corner of a two-dimensional array containing some characters from the initial string and the target string and has a cost value. The cost value in the lower right corner becomes the edit distance value which represents the difference between the two strings. Three types of string operations can be performed by the Levenshtein Distance algorithm, namely: Character Change Operation, Character Add Operation, and Character Delete Operation. A character-changing operation is an operation to swap a character with another character. An example for this case is writing the word "tukae" should become "tukar". In this case, the character "e" is replaced by the letter "r". The character addition operation is an operation adding characters to a string. Adding characters is not only at the end of a word but can be added at the beginning or inserted in the middle of a word. An example for this case is writing the word "belajr" should become "belajar", the character "a" is added in the middle of the word. Character deletion operations are performed to remove characters from a string. For example, in the word "rusaka", the last character is omitted to become the word "rusak". In this operation, the character "a" is deleted.

```
Int LevenshteinDistance (char s[1...m], char t[1...n])
//d is a table with m+1 rows and n+1 columns
Declare int d[0...m, 0...n]
for i from 0 to m
        d[i,0] := i
for j from 0 to n
        d[0,j] := j
for i from 1 to m
        for j from 1 to n
                if s[i] = t[j] then d[i,j] := d[i-1,j-1]
                else
                        d[i,j] := minimum(
                                d[i-1,j] + 1, //deletion
                                d[I,j-1] + 1, //insertion
                                d[i-1,j-1] + 1 ) //substitution
Return d[m,n]
```

Figure 1. Pseudocode of Levenshtein Distance

Methodology

Problems that often occur in typing a document or text are usually caused by a lack of accuracy. Besides, issues arise because the author does not re-check the document or text that has just been completed. After printing a document, a writer needs to re-examine the document. If there are ambiguous or typographical errors, the author has to reprint the document, which causes the use of time to be ineffective. We need to overcome typing errors, and it is necessary to have a way that can make us easier to check whether there are errors in typing text. This spelling correction makes recommendations for the right words according to the word list contained in the dictionary. Therefore, we need an application that automatically provides word recommendations. In this study, the application uses the Levenshtein Distance algorithm to find the distance between 2 strings in a text compared to a list of words in the dictionary.

Design is the initial stage of a series of making an application program. Before creating an application program, the first stage is to design a diagram or chart to describe the process and the final result. Generally, flowchart making is useful to describe the process of compiling an application program. Based on that flowchart, developers can build application programs according to their needs. We can see the flowchart of the word improvement recommendation application program in Figure 2.

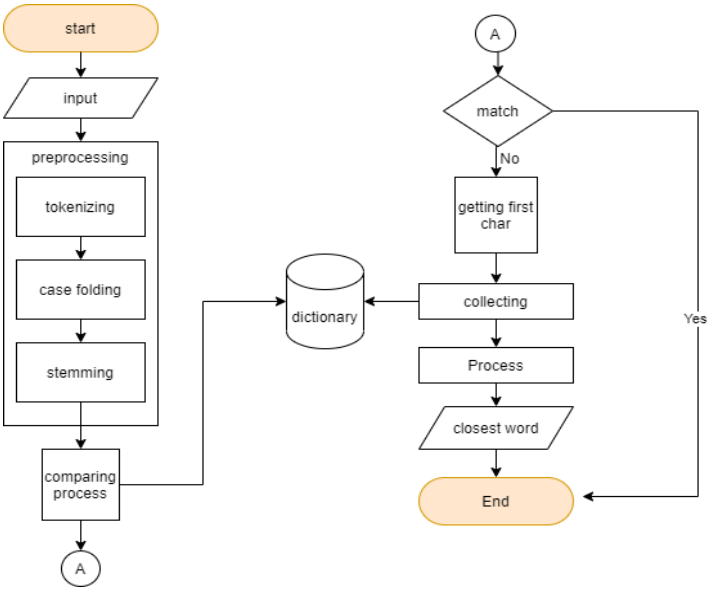

Figure 2. Application flowchart

A context diagram is a Data Flow Diagram used to define the context and system boundaries in modeling. The context diagram includes relationships with entities outside the system. It is often referred to as DFD Level 0 and is the primary determinant of a system modeled in the Data Flow Diagram. The following is a picture of DFD level 0.

Figure 3. DFD level 0

After completing the context diagram, the next step is to detail the primary process in the context diagram. The circle shows an event that needs to be detailed again so that a DFD is formed, which is called a level 1 DFD. Level 1 DFD aims to provide a more in-depth view of the entire system. The existing main process will be broken down into sub-processes. The following is a picture of DFD level 1.

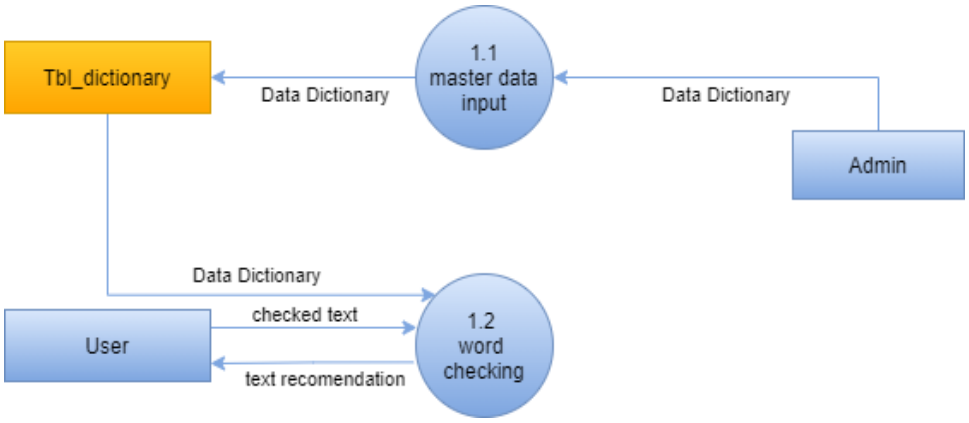

Figure 4. DFD level 1

Designing a good user interface for a system can help users learn, understand, and use it. Creating this user interface is an essential part of implementing a system. This user interface design aims to make it easier for users to use the system to be built [17]. Figure 5 and Figure 6 are the design of a user interface for a recommendation application for correcting words in a text using the Levenshtein distance algorithm.

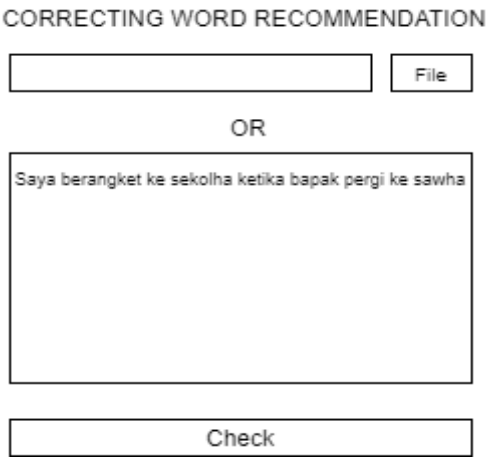

Figure 5. User interface design for the main form

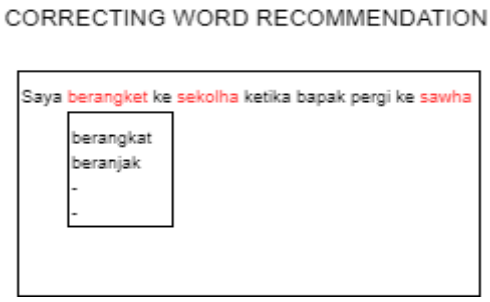

Figure 6. User interface design for recommendation form

When the user presses the Check button, the system will perform the preprocessing process. This preprocessing consists of several parts. These parts are tokenizing, case folding, and stemming. The Tokenizing process is useful for separating or removing input strings based on each constituent word or separating every word composed in a document.

```
//Tokenizing

$tokenizedText = explode(" ",$text);

$totalArrayText = count($tokenizedText);
```

Figure 7. Tokenizing

After running the tokenizing process, the system will run the case folding process. The case folding process is changing the writing of capital letters to lowercase letters in a text.

```
//Case Folding
for($a=0; $a<$totalArrayText; $a++)
{
        $tokenizedText[$a] = strtolower($tokenizedText[$a]);
}
```

Figure 8. Case folding

The last process in the preprocessing stage is stemming. Stemming is a process step that functions to remove prefixes or suffixes to become the root word. After the preprocessing operation is complete, the system will continue the counting process with Levenshtein Distance. The system will display the results in the form of word recommendations based on the Levenshtein Distance calculation results.

Results and Discussions

Testing of this recommendation application is carried out based on the user's input, which is a combination of letters, numbers, and punctuation marks. Users freely enter words, numbers, and punctuation marks. But the app checks the spelling of the word only and shows the closest recommended term. The following testing can be seen in Figure 9:

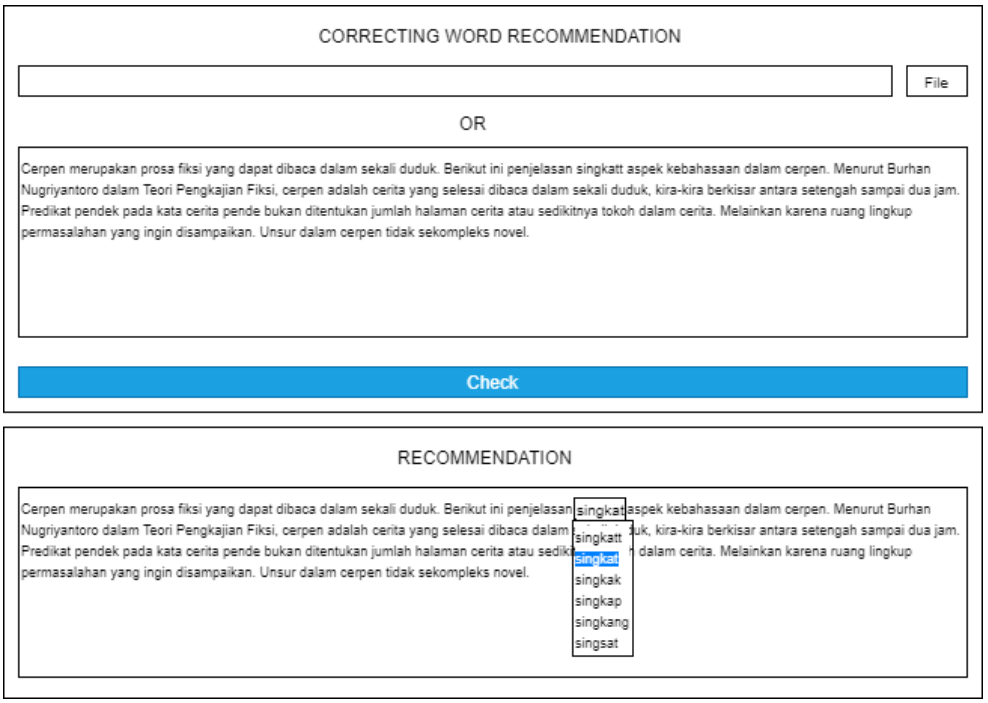

Figure 9. Testing

The following is a formula for calculating the level of accuracy of the word improvement recommendations. The formula is shown in Equation (1) below.

$$accuracy = \frac{number\ of\ words\ correct}{total\ word\ count} \times 100\% \quad (1)$$

This word improvement recommendation application was tested on ten different texts. The first was a text consisting of <100 words, the second was a text consisting of 100 to 250 words, and the third was a text consisting of 250 to 500 words. The following is a recapitulation of the average calculation results.

Results

Results should be clear and concise. The results should summarize (scientific) findings rather than providing data in great detail. Results of data analyses can be presented in tables, graphs, figures, or any combination of the three. The authors are advised to use proper variation in presenting tables, graphs, or verbal descriptions. All displayed tables and graphs should be referred to in the text. What answer was found to the research question; what did the study find? Was the tested hypothesis true?. The level of accuracy can be seen in Table 1 below.

Table 1. The level of accuracy

Number of Words	Accuracy
<100	95%
100 – 250	94%
250 - 500	90%

Tests were carried out 30 times with 30 different data. The first ten texts are less than 100 words long, the following ten texts are 100 to 250 words long, and the last ten texts are 250 to 500 words long. Table 1 shows the average level of accuracy of the above tests. The test results on these three conditions showed that the accuracy value is equally high, more than or equal to 90%. This fact means that the Approximate String-Matching method, which in this study uses the Levenshtein Distance algorithm, has a good level of accuracy. These results are in line with Braddley's statement, which states that the Approximate String Matching method has a

reasonably good level of accuracy, performance, approximate approach, and metric MAP (Mean Average Precision) value [3]. In other applications, the correction system is based solely on a dictionary or a collection of vocabulary words stored in a file. The solution to the problem offered in this study is the use of a table that contains a collection of standard words and non-standard words. Here, when a word is not found in the dictionary, it will be checked based on the standard word table. This case is also one of the reasons why the accuracy percentage is not high.

Conclusion

Based on the discussion and analysis results of the development of the Word Improvement Recommendation Application using the Levenshtein Distance Algorithm, we can conclude that:

a) The word improvement recommendation application using the Levenshtein Distance algorithm has an average accuracy rate of 95% for text <100 words, whereas for text with 100-250 words it has an accuracy rate of 94%, and for text with 250-500 words it has an accuracy rate of 90%.
b) The downside of this application is that this application still cannot detect errors for incorrect affixes and cannot compare two words with a different number of characters.

Acknowledgments

Thank you to the editors and reviewers of the Journal of Technology and Information Management for their willingness to process this article.

References

[1] Y. P. Putri and R. Lawson, "Aplikasi pengkoreksi kesalahan ejaan dan padanan kata pada tugas akhir mahasiswa," *Inform. Mulawarman J. Ilm. Ilmu Komput.*, vol. 14, no. 2, pp. 72–75, 2019.

[2] E. S. Nasution, N. A. Hasibuan, and S. Suginam, "Implementasi algoritma approximate string matching pada aplikasi filosofi berbasis android," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 2, no. 1, pp. 466–470, 2018.

[3] M. O. Braddley, M. Fachrurrozi, and Y. Novi, "Pengoreksian ejaan kata berbahasa Indonesia menggunakan Algoritma Levensthein Distance," in *Prosiding Annual Research Seminar*, 2017, vol. 3, no. 1, pp. 167–171.

[4] M. R. Qhadafi, "Analisis kesalahan penulisan ejaan yang disempurnakan dalam teks negosiasi siswa SMA Negeri 3 Palu," *J. Bhs. dan Sastra*, vol. 3, no. 4, pp. 1–21, 2018.

[5] A. I. Fahma, "Identifikasi kesalahan penulisan kata (typographical error) pada dokumen berbahasa Indonesia menggunakan metode N-gram dan Levenshtein Distance," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 1, pp. 53–62, 2018.

[6] B. V. Indriyono, "Kombinasi Damerau Levenshtein dan Jaro-Winkler Distance untuk koreksi kata bahasa Inggris," *J. Tek. Inform. dan Sist. Inf.*, vol. 6, no. 2, pp. 162–173, 2020.

[7] W. S. Ismail, P. W. Purnawan, I. Riyanto, and N. Nazori, "Sistem perekaman pelat nomor mobil pada palang pintu parkir menggunakan web kamera dan mikrokontroler," *Matrix Jurnal Manajemen Teknologi dan Informatika*, vol. 10, no. 3, pp. 103–112, 2020.

[8] N. H. Ariyani, Sutardi, and R. Ramadhan, "Aplikasi pendeteksi kemiripan isi teks dokumen menggunakan metode Levenshtein Distance," *semanTIK*, vol. 2, no. 1, pp. 279–286, 2016.

[9] L. Hermawan and M. Bellaniar Ismiati, "Pembelajaran text preprocessing berbasis simulator untuk mata kuliah information retrieval," *J. Transform.*, vol. 17, no. 2, pp. 188–199, 2020.

[10] A. I. Kadhim, "An evaluation of preprocessing techniques for text classification," *Int. J. Comput. Sci. Inf. Secur.*, vol. 16, no. 6, pp. 22–32, 2018.

[11] J. Joseph and J. R. Jeba, "Information extraction using tokenization and clustering methods," *Int. J. Recent Technol. Eng.*, vol. 8, no. 4, pp. 3690–3692, 2019.

[12] S. Suprianto, S. Sunardi, and A. Fadlil, "Aplikasi sistem temu kembali angket mahasiswa menggunakan application of information retrieval for opinion student," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 6, no. 1, pp. 33–40, 2018.

[13] A. Rasool, A. Tiwari, G. Singla, and N. Khare, "String matching methodologies: A comparative analysis," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 2, pp. 3394–3397, 2012.

[14] F. Friendly, "Jaro-Winkler Distance improvement for approximate string search using indexing data for multiuser application," in *Journal of Physics: Conference Series*, 2019, vol. 1361, no. 1, pp. 1–7.
[15] K. Al-Khamaiseh and S. Alshagarin, "A survey of string matching algorithms," *J. Eng. Res. Appl.*, vol. 4, no. 2, pp. 144–156, 2014.
[16] M. M. Yulianto, R. Arifudin, and A. Alamsyah, "Autocomplete and spell checking levenshtein distance algorithm to getting text suggest error data searching in library," *Sci. J. Informatics*, vol. 5, no. 1, pp. 67–75, 2018.
[17] I. Astrina, M. Z. Arifin, and U. Pujianto, "Penerapan algoritma FP-Growth dalam penentuan pola pembelian konsumen pada kain tenun medali mas," *Matrix Jurnal Manajemen Teknologi dan Informatika*, vol. 9, no. 1, pp. 32–40, 2019.