Improvement Of Query Speaking on The Indonesian to Madura Dictionary Using Levenshtein Distance Method

M. Yahya Ubaidillah[1], Muchamad Kurniawan[1*], Septiyawan Rosetya Wardhana[1]
[1]Informatics Engineering, Adhi Tama Institute of Technology, Surabaya, Indonesia
e-mail: muchamad.kurniawan@itats.ac.id
*Corresponding author

ABSTRACT

Men are distinguished from other living beings by their use of language, which becomes one of their most distinctive and humanistic qualities. Many different languages are spoken worldwide, including Indonesian, which has approximately 742 different dialects. Due to the unique language of Madura, which is located on a large island with numerous beach tourism destinations, tourists will have difficulty navigating the island. People outside Madura Island who come to visit or vacation will find it difficult to communicate with the locals during their stay or holiday. An Indonesian to Madurese translation dictionary is therefore required in this case. The Levenshtein Distance method was employed in this investigation. The algorithm in the dictionary is used to process the search for the closest distance (dif) between the words being inputted and the words that are already in the database. To provide a prototype for the use of dictionaries. Indonesian and Madurese data sets were used in the investigation by the researcher. According to the simulation results acquired after multiple trials, the error accuracy was 90 % for the first letter input, 84 % for the middle letter input, and 84 % for the last letter input for the first letter. As a result, according to the study's findings, the accuracy of this dictionary increased by 86 %. The first letter received 90 % of the votes, the middle letter received 84 %, and the last letter received 84 %. As a result, according to the study's findings, the accuracy of this dictionary increased by 86 %. The first letter received 90 % of the votes, the middle letter received 84 %, and the last letter received 84 %. As a result, according to the study's findings, the accuracy of this dictionary increased by 86 %.

Keywords: Indonesian to Madurese translation, dictionary, Levenshtein Distance Algorithm

Article History		
Received : Nov, 10rd 2021	Revised : Des, 2nd 2021	Accepted : Des, 4th 2021

I. INTRODUCTION

Language is a mode of communication used to express one person's goals, ideas, feelings, and thoughts to another through another. We can communicate with others without trouble because of the language we speak. On the other hand, individuals will find it difficult to communicate their desires if they do not speak the language of the situation. [1] As a result, acquiring and continually improving one's language skills is essential. The Java Archipelago has its language, especially in Indonesia. Madura Island is the largest archipelago in Indonesia as a whole. Madura Island has a regional language that is used for daily communication. The Madurese language is different from other languages, and it will be difficult for those traveling or vacationing outside of Madura to converse. The automatic correction of spelling mistakes is one of the most significant developments in Natural Language Processing (NLP). The Levenshtein Distance / Edit Distance approach is one of the decision-making procedures used in this study line, dating back to the 1960s of the previous centuries [2]. For this reason, it is also known as the string-matching method. It can detect differences between two words by comparing their strings.

Testing on the System Song Finder by Lyrics with Command Voice in the Song Search System Application research utilizing the Levenshtein Algorithm has been done to understand the algorithm better. The algorithm developed by Levenshtein can be used to compute distance. Users can use this strategy to locate song titles based on a few lyrics and receive the most relevant search results possible. The Levenshtein algorithm uses the additional mechanism, character insertion, and deletion to accomplish its goals. It is possible to employ distance to address the problem of misspelled phrases [3], although this is not recommended. Detecting similarities between two strings that could be used to plagiarize is accomplished using the Levenshtein distance, which is a technique. In previous research, the Levenshtein Distance Algorithm technique was found to be capable of converting 100 % of terms in tweets with spelling errors into keywords for issue category categories [4].

The search module in an e-dictionary is critical for increasing user participation in searching. Results The e-dictionary module's search feature makes use of the Levenshtein Distance algorithm. Autocorrect validation and autocorrection can identify a term typing error and then make suggestions to produce the correct results automatically. The auto-correction function provides

DOI: 10.25139/ijair.v3i2.4394

an output based on the drug words entered and finds the drug term closest to the database. [5] The results of evaluating the algorithm's implementation on the medicine e-dictionary were used to calculate the accuracy value, recall, and precision of 90 % [6]. According to the study "Implementation of the Autocomplete Feature and Levenshtein Distance Algorithm to Improve the Effectiveness of Word Search in the Big Indonesian Dictionary (*KBBI*)," the effectiveness test of the implementation of autocomplete in applications scored 84.615 %, indicating that this feature is very effective. In addition, the Levenshtein distance accounts for 76.04 % of the overall distance traveled. According to this, it is appropriate for usage in *KBBI* applications [6]. If you are comparing methods for measuring the distance between words in strings, you should compare the results of your system with those of experts to find out how accurate your system is at what it is doing.

The testing results indicate that Levenshtein outperforms Jaro Winkler in overall performance. According to the test results, the Levenshtein technique has an accuracy value of 46.15 %. According to the data, the Jaro Winkler technique has an accuracy rate of 38.46 %. Jaro Winkler's method is inferior to the Levenshtein method, as demonstrated by this example. These two methods can only be used to find almost identical words save for a few minor spelling errors. These two strategies are incapable of dealing with words that have similar meanings. Developing a system for determining related phrases based on the meaning of a word is one of the potential future research directions. [7]. On the most basic level, database queries can be used to implement the dictionary application. On the other hand, Database queries contain several limitations that make the dictionary less than ideal, such as the fact that when a user enters in the wrong word, the dictionary translation process cannot be resumed and/or results are not obtained. As a result, the Levenshtein Distance technique is required to generate a dictionary that is both faster and more accurate than before. The accuracy of the search results is improved by employing this algorithm. Even if the user enters an inaccurate term, this algorithm will still be able to locate the information they require and provide search options related to the word they typed [8]. According to prior research, the Levenshtein distance algorithm achieves satisfactory results when matching string data to provide text suggestions, such as in handwriting recognition, search words, and misspelled words, thereby increasing the effectiveness of input, avoiding spelling errors, and speeding up human-computer interactions [9, 10].

II. METHOD

This system flow illustrates the flow of the system work process to make the analysis process more efficient and effective. As illustrated in Fig. 1, where:

- Input is where the user types in a word in Indonesian.
- In addition, by assessing the resemblance or similarity between two words, the Levenshtein Distance will allow the system to improve the basic words.
- The Translation Results section shows that it will be translated into Madurese only if the word is correct.

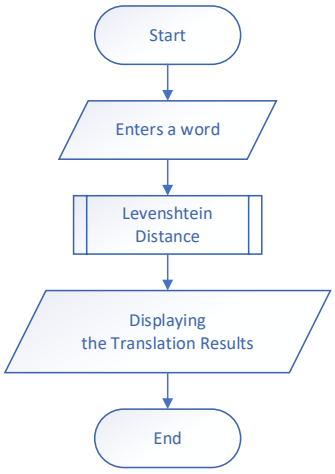

Fig. 1. System Flow

It is possible to measure the difference or distance between two strings by utilizing the Levenshtein distance, a string matrix, as a measuring tool. The smallest value possible based on the number of change operations required to transform one string into another string is employed to determine the distance between these two strings. In addition to the three operations [5,] there is the operation of insertion, deletion, and the exchange operation. Equation (1) is used to get the value of similarity after acquiring the cost of edit-Similarity has been determined [6].

$$Sim = 1 - \frac{diff}{maxCS,ST} \times 100 \quad (1)$$

DOI: 10.25139/ijair.v3i2.4394

The Levenshtein Distance Algorithm function is tested in the translated case with the input string and the target string "NOT".

TABLE I
LEVENSHTEIN DISTANCE CALCULATION

	-	B	u	k	a	n
-	0	1	2	3	4	5
B	1	0	1	2	3	4
u	2	1	0	1	2	3
k	3	2	1	0	1	2
u	4	3	1	1	1	2

For the calculation of the distance using the Levenshtein distance, it is necessary to compute the similarity value measurement to determine how much the degree of similarity is between the two tables in Table I based on Equation (2).

$$'Sim = (1 - \frac{2}{5}) \text{ X } 100 \quad (2)$$
$$= 100 \, X \, 0{,}6$$
$$= 60$$

A. Data Preparations

Table II contains the data gathered for this study from the Madura Dictionary from the regional library and an interview procedure with Madura language culturalists. Approximately 28527 Indonesian basic words and 500 Madurese words will be used as a dataset in this study. The data is Madurese vocabulary data that will be utilized as a dataset in this study.

TABLE II
DATASETS SAMPLE

Madurese	Indonesian
	A
aba [aba] {Ar.}	*bapak*
aba' [abe?]	*diri*
abad [a.bad]	*abad (masa serratus tahun)*
aban [aben]	*siang (antara pukul 10.00 – 12.00)*
abit [abit]	*lama*
	B
ba'a [be'e]	*banjir*
ba'en [be'en]	*kamu*
babine' [bebine']	*perempuan*

B. Testing Scenario

A test is performed on the Levenshtein Distance Algorithm function to determine the degree of correctness of word changes in input words compared to words in the database. The results of evaluating all the functions of the dictionary application can be used to conclude the application.

Based on the data set, four types of tests were carried out. Experiment with wrong word input. When comparing two types of words that have structural errors, this test is intended to determine the level of accuracy of the Levenshtein distance method. Sample words that can be used, for example:

- *Baku - Buku*
- *Buqu - Buku*
- *Buki - Bukan*
- Double G(gg)

Following the application of the Levenshtein Distance method, it can be inferred that, based on the findings of the similarity values in each test, the test with the highest similarity value is *Baku - Buku* (85).

III. RESULT AND DISCUSSION

There is a summary of the outcomes of the trials and a discussion of the procedure for computing the Levenstein Distance Algorithm in this section of the document. Following the design process described in the previous chapter, the next step is to conduct a trial run to determine the level of success of the method used, and the implementation of the method following the

design developed previously to draw conclusions and make recommendations for further research. As shown in Table III, by inputting incorrect prefixes of numerous words, such as *ejak*, *ikhir*, *olat*, *qneh*, *ei*, *cotak*, and *umpat* construct terms that were similar to what the user had expected. The words *betik*, *babar*, *enpat*, *tilanga*, and *ahas* are not the words expected by the user; however, after testing with a different prefix, we find 45 correct input words and five incorrect input words. From 50 test words, we can calculate the curation value, accuracy = 45/50 x 100 % = 90 %, as depicted in Figure 2.

TABLE III
WRONG FIRST LETTERS INPUT

No	Incorrect Front Letter Input	Word Repair	Expected word	Appropriate/Not Appropriate
1	*Ejak*	*Ajak*	*Ajak*	Appropriate
2	*Ikhir*	*Akhir*	*Akhir*	Appropriate
3	*Olat*	*Ulat*	*Ulat*	Appropriate
4	*Qneh*	*Aneh*	*Aneh*	Appropriate
5	*Betik*	*Detik*	*Batik*	Not Appropriate
6	*Ei*	*Di*	*Di*	Appropriate
7	*Babar*	*Sabar*	*Bubar*	Not Appropriate
8	*Cotak*	*Botak*	*Botak*	Appropriate
9	*Umpat*	*Empat*	*Empat*	Appropriate
10	*Euri*	*Duri*	*Duri*	Appropriate
11	*Empat*	*Ingat*	*Empat*	Not Appropriate
12	*Poto*	*Foto*	*Foto*	Appropriate

As shown in Table IV, the test that came from incorrectly typing the middle letter of various words, including *axir*, *otat*, *ameh*, *bawik*, *buhar*, and *bovak*, yielded terms identical to the user had expected to find. The words *atak*, *ie*, *gatil*, *gembut*, *ipi*, and *alak* do not correspond to the words expected by the user. After testing the incorrect prefix, 42 correct input words and eight incorrect input words are found. Fifty test words, we can calculate the curation value as accuracy = 42/50 x 100 % = 84 %, as depicted by the graph in Figure 2.

TABLE IV
WRONG MIDDLE LETTER INPUT

No	Wrong Middle Letter Input	Word Repair	Expected word	Appropriate/Not Appropriate
1	*Atak*	*Otak*	*Ajak*	Not Appropriate
2	*Aksir*	*Akhir*	*Akhir*	Appropriate
3	*Otat*	*Ulat*	*Ulat*	Appropriate
4	*Ameh*	*Aneh*	*Aneh*	Appropriate
5	*Bawik*	*Betik*	*Batik*	Appropriate
6	*Ie*	*Ke*	*Ke*	Not Appropriate
7	*Buhar*	*Bubar*	*Bubar*	Appropriate
8	*Bovak*	*Botak*	*Botak*	Appropriate
9	*Empt*	*Empat*	*Empat*	Appropriate
10	*Dri*	*Tari*	*Dari*	Appropriate
11	*Enpat*	*Ingat*	*Empat*	Not Appropriate
12	*Foso*	*Foto*	*Foto*	Not Appropriate

It is explained in Table V that the test that resulted from incorrectly entering the back letter of various words such as *ajac*, *akhit*, *ulax*, *unej*, *batik*, and do result in terms that were identical to what the user had anticipated. The words *fota*, *haud*, *jaub*, *juab*, *mang*, and *kama* do not correspond to the words expected by the user. After testing the incorrect prefix, there are 42 correct input words and 8 incorrect input words. Fifty test words, we can count the curation value as accuracy = 42/50 x 100 % = 84 %, as depicted by the graph in Figure 2.

TABLE V
WRONG BACK LETTER INPUT

No	Wrong Back Letter Input	Word Repair	Expected word	Appropriate/Not Appropriate
1	*Ajac*	*Ajak*	*Ajak*	Appropriate
2	*Akhit*	*Akhir*	*Akhir*	Appropriate
3	*Ulax*	*Ulat*	*Ulat*	Appropriate
4	*Anej*	*Aneh*	*Aneh*	Appropriate
5	*Batic*	*Detik*	*Batik*	Appropriate
6	*Do*	*Ke*	*Ke*	Appropriate
7	*Bubay*	*Bubar*	*Bubar*	Appropriate
8	*Botat*	*Botak*	*Botak*	Appropriate
9	*Empay*	*Empat*	*Empat*	Appropriate
10	*Duro*	*Duri*	*Duri*	Appropriate
11	*Empax*	*Empat*	*Empat*	Appropriate
12	*Fota*	*Kota*	*Foto*	Not Appropriate

DOI: 10.25139/ijair.v3i2.4394

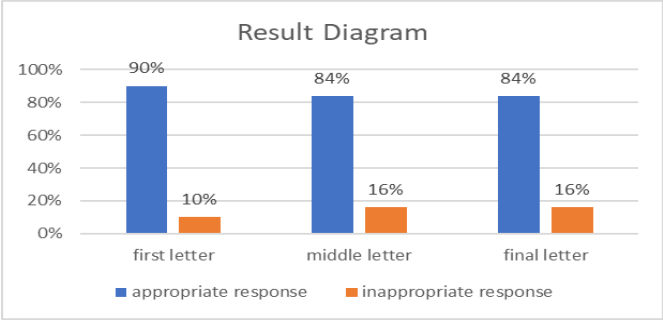

Fig. 2. Diagram of Input Test Result Diagram Letters

TABLE VI
THE SUMMARY OF TESTING RESULTS

Letter error test	Appropriate Category	Not Appropriate Category	The accuracy Method
first letter	45	5	90
middle letter	42	8	84
final letter	42	8	84
		Total:	86%

IV. CONCLUSION

It is possible to apply the Indonesian to the Madurese translation dictionary using the Levenshtein Distance Method even if the user enters the incorrect word in the input field. When two words have the same similarity value, the leading variable error takes precedence over the other words in the calculation. The testing procedure was carried out in three trials. The results showed that the percentage of front input errors was 90 % accurate, the percentage of middle letter input errors was 84% accurate, the percentage of back letter input errors was 84 % accurate, and the percentage of double g input errors was 96 % accurate. According to the description in Table 6, with a total accuracy of 86 %.

REFERENCES

[1] Rina Devianty (2017) 'Bahasa Sebagai Cermin Kebudayaan', Jurnal Tarbiyah, 24(2), pp. 226–245D. Rahmayanti and I. Pendahuluan, "Optimasi Routing Berbasis Algoritma Genetika Pada Sistem Komunikasi Bergerak," *J. Electr. Electron. Commun. Control. Informatics, Syst.*, vol. IV, no. 1, pp. 18–23, 2010.

[2] Kukich, K., (1992) Techniques for Automatically Correcting Words in Text, ACM Computing Surveys, Volume 24, No.4, pp, 377-439, December.

[3] H. T. Saidah, M. S. N. Ishlah, and N. N. Rokhmah, "Autocorrect pada Modul Pencarian Drugs e-Dictionary Menggunakan Levenshtein Distance," Rekayasa Sist. dan Teknol. Inf., vol. 4, no. 1, pp. 64–69, 2020, [Online]. Available: https://doi.org/10.29207/resti.v4i1.1401.

[4] Rosmala, D. and Risyad, Z. M. (2018) 'Algoritma Levenshtein Distance dalam Aplikasi Pencarian isu di Kota Bandung pada Twitter', MIND Journal, 2(2), pp. 1–12. doi: 10.26760/mindjournal.v2i2.1-12.

[5] Kukich, K., (1992) Techniques for Automatically Correcting Words in Text, ACM Computing Surveys, Volume 24, No.4, pp, 377-439, December.

[6] Umar, R., Hendriana, Y., & Budiyono, E. (2015). Implementation ofLevenshtein Distance Algorithm for E-Commerce of Bravoisitees Distro.International Journal of Computer Trends and Technology (IJCTT), 27 (3), 131-136.

[7] Putra, M. E. W., & Suwardi, I. S. (2015). Structural off-line handwriting character recognition using approximate subgraph matching and levenshtein distance. Procedia Computer Science, 59, 340-349.

[8] Nisa, K. and Ngafidin, M. (2015) 'Implementasi Fitur Autocomplete dan Algoritma Levenshtein Distance untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia (KBBI)', Jurnal Teknik Elektro, 7(1), pp. 1–6. doi: 10.15294/jte.v7i1.8578.

[9] Fauzan, R., Riadi, J. and Sholihin, F. (2018) 'Perbandingan Metode Perhitungan Kemiripan', Prosiding SNRT (Seminar Nasional Riset Terapan), 5662(November), pp. 1–6.

[10] Fauzan, R., Riadi, J. and Sholihin, F. (2018) 'Perbandingan Metode Perhitungan Kemiripan', Prosiding SNRT (Seminar Nasional Riset Terapan), 5662(November), pp. 1–6.

[11] Ariyani, N. H., Sutardi and Ramadhan, R. (2016) 'Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance', semanTIK, Vol 2(1), pp. 279–286. Available at: http://ojs.uho.ac.id/index.php/semantik/article/download/1030/661.