Scientific Journal of Informatics
Vol. 9, No. 1, May 2022

p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Performance Comparison of Similarity Measure Algorithm as Data Preprocessing Stage: Text Normalization in *Bahasa Indonesia*

Achmad Yohni Wahyu Finansyah[1], Afiahayati[2*], Vincent Michael Sutanto[3]
[1,2]Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia
[3]Institute for Research Initiatives / Division of Information Science, Nara Institute of Science and Technology, Nara, Japan

Abstract.
Purpose: More and more data are stored in text form due to technological developments, making text data processing more difficult. It also causes problems in the text preprocessing algorithm, one of which is when two texts are identical, but are considered distinct by the algorithm. Therefore, it is necessary to normalize the text to get the standard form of words in a particular language. Spelling correction is often used to normalize text, but for *Bahasa Indonesia*, there has not been much research on the spell correction algorithm. Thus, there needs to be a comparison of the most appropriate spelling correction algorithms for the normalization process to be effective.
Methods: In this study, we compared three algorithms, namely Levenshtein Distance, Jaro-Winkler Distance, and Smith-Waterman. These algorithms were evaluated using questionnaire data and tweet data, which both are in *Bahasa Indonesia*.
Result: The fastest normalization time is obtained by the Jaro-Winkler, taking an average of 31.01 seconds for questionnaire data and 59.27 seconds for tweet data. The best accuracy is obtained by the Levenshtein Distance with a value of 44.90% for the questionnaire data and 60.04% for the tweet data.
Novelty: The novelty of this research is to compare the similarity measure algorithm in *Bahasa Indonesia*. Therefore, the most suitable similarity measure algorithm for *Bahasa Indonesia* will be obtained.

Keywords: Preprocessing, Text Normalization, Spell-correction, Levenshtein Distance, Jaro-Winkler Distance, Smith-Waterman
Received May 2021 / **Revised** June 2021/ **Accepted** February 2022

INTRODUCTION

The growth of the industry in the technology sector has made stored data bigger and made it difficult to process data. This makes research in the field of text mining continue to develop [1]. In line with this, new problems in text mining have also been revealed, such as large amounts of data, high dimensionality, various data structures, and noise in data [2]. This problem, specifically unstructured data, makes the data inconsistent and causes the results of natural language processing to be inaccurate [3]. One of the causes of unstructured data is Non-Standard Words (NSWs), which is a condition when words cannot be found in a dictionary. Therefore, one cannot find a specific word in the word list and also cannot derive the morphological meaning of the words from the dictionary [4]. The solution to this problem is text normalization. Text normalization is used at the data preprocessing stage, helping to remove or replace informal writing or NSWs into its standard form in language [4]. Text normalization can be done in various ways, including removing punctuation marks, changing capitalization, spell correction, and adding, deleting, or rearranging words [5].

The implementation of spelling correction usually uses the word that has the closest distance or highest similarity measure to a word in the dictionary. The algorithms for similarity measurement are divided into several categories, such as edit based (Levenshtein Distance, Jaro-Winkler Distance, Hamming Distance), token-based (n-gram), domain-dependent, and hybrid (TF-IDF) [3]. Comparison of these algorithms has

[*] Corresponding author.
Email addresses: achmad.y@mail.ugm.ac.id (Achmad), afia@ugm.ac.id (Afiahayati), Vincent (v.michael@mail.ugm.ac.id)
DOI: [10.15294/sji.v9i1.30052](https://doi.org/10.15294/sji.v9i1.30052)

been done before. Nugraha [6] conducted a comparison between the Longest Common Subsequence algorithm with Levenshtein Distance and Jaro-Winkler Distance.

The results showed that normalization using the Levenshtein Distance and Jaro-Winkler method had better accuracy than LCS. [3] conducted a comparison of Jaro-Winkler Distance and Smith-Waterman in detecting duplicate data in the English dataset. The experiment shows that Smith-Waterman matches strings with more accurate results than Jaro-Winkler Distance. Research on normalization has also been carried out for English, Arabic, French, and Burmese language [7], [8], [9].

These previous works show that text normalization usually uses Levenshtein Distance or Jaro-Winkler Distance for spelling correction. Besides, other work shows that Smith-Waterman provides higher accuracy than Jaro-Winkler Distance in detecting duplicate data in British medical datasets [3]. However, there has never been any research comparing the performance of these algorithms on text normalization for the *Bahasa Indonesia* dataset. Therefore, a comparison of the normalization algorithm for the *Bahasa Indonesia* dataset is needed.

Based on the background, this research aims to compare Levenshtein Distance, Jaro-Winkler Distance, and Smith-Waterman as a similarity measure algorithm for text normalization, specifically in *Bahasa Indonesia*. The parameters which will be compared are accuracy and time needed for normalizing. By choosing an algorithm suitable for the *Bahasa Indonesia* language structure, it is expected to reduce the time of execution and increase the accuracy when used in natural language processing. This study is an extended version of the authors' thesis [10].

METHODS

In this section, all steps used in this study are explained. First, is the dataset used. The dataset used consists of two different types of data, which are formal-writing-style and informal-writing-style dataset. The dataset was preprocessed using several techniques in the following order: data cleansing, case folding, and normalization. Afterwards, the preprocessed data were spell-corrected using three distinct methods: the Levenshtein Distance, the Jaro-Winkler Distance, and the Smith-Waterman method. During the spell-correction process, the time and performance of each method were noted. The performance of each method was then compared, followed by a discussion on which is the best method to be used in the scope of formal and informal *Bahasa Indonesia*. The research stages are visualized in Figure 1.

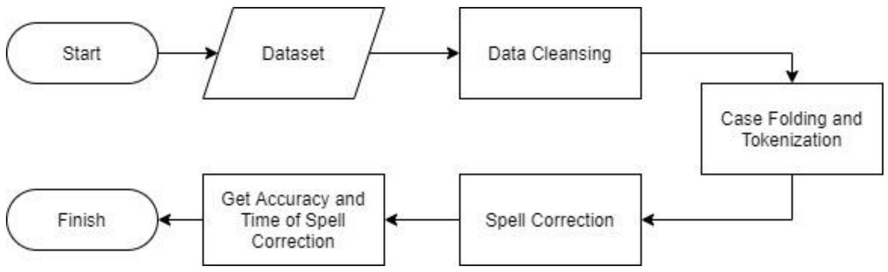

Start
Dataset
Data Cleansing
Case Folding and Tokenization
Spell Correction
Get Accuracy and Time of Spell Correction
Finish

Figure 1. Research stages

Dataset

The research will use 2 kinds of datasets in *Bahasa Indonesia*. The first dataset was taken from the questionnaire of student's comments on the teacher. Thus, the language is more formal. The second dataset was acquired through web-scraping Indonesian tweets. This data has more free writing style and more NSW compared to the first dataset. Each dataset consists of 1500 data rows written in *Bahasa Indonesia*. A subset of the questionnaire and tweet dataset are presented in Table 1 and Table 2, respectively.

Tabel 1. Subset of the questionnaire dataset

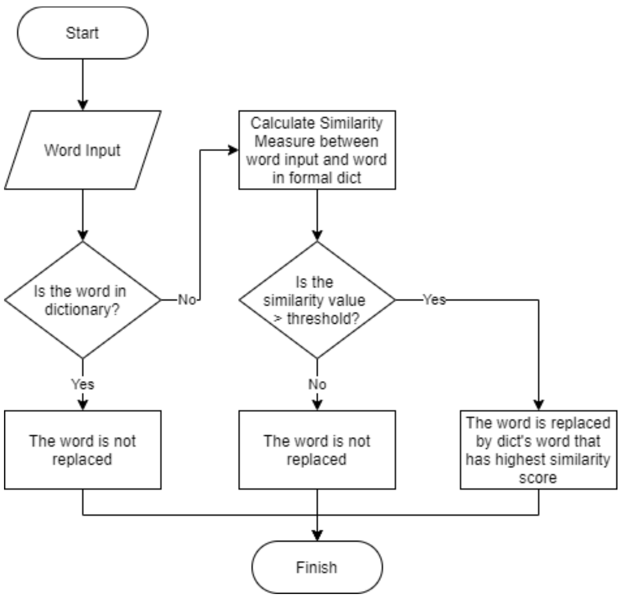
No	Questionnaire Answers
1	Guru idaman sepanjang masa
2	Tetaapp baik, ramah, dan sabarr ya bu
3	Ibu terbaikkk!!
4	Semoga pembelajaran menjadi semakin baik dsan efektif.
5	Lebih dijelaskan lagi bagian yang sulit

Tabel 2. Subset of the tweet dataset

No	Tweets
1	b'@p_genedi Wah Mantep nih Ruangguru semakin Berinovasi dengan menu baru \xf0\x9f\xa5\xb0'
2	"b'@edcfess2 Mulai kelas 8 nyicil2.. buat un smp aku ga bimbel, Cuma ngandelin pt di sekolah sama zenius aja wkwk. Alhamdulillah dapet 36 \xf0\x9f\x98\xab"
3	b'@anakcangtip terimakasih @ruangguru_ berkatnya aku beres ujian pertama xixi'
4	"b'zenius error yaa? Aku bukaa di web kok ""hljsj lite: network error"" teruss… helpp"
5	"b'mau lo quipper, zenius, ruangguru, brainly adalah jalan terbaique. https://t.co.2aBBJgcuXk"

Data Cleansing, Case Folding, and Tokenization
Before the normalization process, the data must be cleaned to reduce data variance. The data cleansing stage uses regular expressions to remove punctuation marks, symbols, and URLs, which often occurred specifically in the tweet dataset. This had to be done as URL and symbols have no contribution in the future process. Afterwards, the case folding and tokenization processes are carried out. The tweets will be converted at the case folding stage by lowercasing every letter. By lowercasing each letter, we created a well-uniformed dataset. At the tokenization stages, every word that exists within a sentence is tokenized into a list. This stage will facilitate the spell-correction stage, where when making spelling corrections, the input received must be in the form of a single word and not sentences.

Spell Correction
The spell correction method is important when dealing with vast user inputs, in this case, users' tweets. Thus, several similarity measure algorithms will be used and compared. The algorithms that will be compared are Levenshtein Distance, Jaro-Winkler Distance, and Smith-Waterman. The process of spell correction can be seen in Figure 2.


Start
Word Input
Is the word in dictionary?
No
Yes
Calculate Similarity Measure between word input and word in formal dict
Is the similarity value > threshold?
Yes
No
The word is not replaced
The word is not replaced
The word is replaced by dict's word that has highest similarity score
Finish
Figure 2. Spell correction scheme

The scheme of the spelling correction process starts with checking word-by-word input with the words in the dictionary. The dictionaries used are the slang dictionary and the formal dictionary. The slang dictionary is a collection of informal words. Like other languages, Indonesian has several forms of everyday language [11]. Examples of slang dictionaries can be seen in Table 3. A formal dictionary is a collection of formal words. This dictionary is used to determine which words should be normalized by common words in the dictionary. Examples of formal dictionaries can be seen in Table 4.

Tabel 3. Slang dictionary

Slang	Formal
Met	Selamat
Netas	Menetas
Nyenengin	Menyenangkan
Udah	Sudah
Gitu	Begitu

Tabel 4. Formal dictionary

Formal
abad
abadi
abal
abar
abdi

If a word cannot be found in both dictionaries, then it is necessary to correct the spelling. Spelling correction is done by calculating the similarity of words to each word in a formal dictionary. The most similar dictionary's word will replace the word in the dataset. In this stage, three algorithms will be used to calculate the similarity. The algorithm that will be compared is Levenshtein Distance, Jaro-Winkler Distance, and Smith-Waterman.

Levenshtein Distance

Levenshtein Distance, proposed by Vladimir Levenshtein in 1965, is an algorithm for calculating the distance between two texts or groups of characters [12]. This distance is calculated based on the minimum number of transformations of a string into another string, which includes deletion, insertion, and replacement [13]. The Levenshtein distance between two strings a, b (which have lengths $|\ a\ |$ and $|\ b\ |$) can be defined as (1).

$$lev_{a,b}(i,j) = \begin{cases} max(i,j), \text{ if } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a\neq b)} \end{cases} \text{otherwise} \end{cases} \tag{1}$$

Where $1_{a\neq b}$ is equal to 0 when $a_i = b_j$ and equal to 1 otherwise.

Jaro-Winkler Distance

The Jaro-Winkler process proposed by William E. Winkler is refined by the Jaro Distance algorithm [14]. Jaro's distance algorithm determines the similarity value of two words by counting the number of characters that correspond to the two words that are not too far away and reducing the number of characters that match up to half of the number of characters undergoing transposition. The similarity value of the Jaro Distance algorithm can be calculated by (2), where $|s_i|$ is Length of the string s_i, m is the number of macthing characters, and t is the half of the number of transpositions [15].

$$sim_j = \begin{cases} 0, m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), m \neq 0 \end{cases} \tag{2}$$

The characters in s_1 and s_2 is declared match if the position difference in certain position is not more than:

$$\left|\frac{max(|s_1|\cdot|s_2|)}{2}\right| - 1 \tag{3}$$

Winkler added the application of penalties for inappropriate characters in the first four characters in the Jaro Distance algorithm. The Jaro-Winkler similarity value is defined as follows:

$$sim_w = sim_j + l\,p(1 + sim_j) \tag{4}$$

Where sim_j is the Jaro similarity, l is the length of common prefix (capped at 4), and p is a constant scaling factor for how much the score is adjusted upwards for having common prefixes.

Smith-Waterman

The Smith-Waterman algorithm is an algorithm used to compare two nucleotide sequences or protein structures in the field of bioinformatics. By applying the sequence alignment function of the Smith-Waterman algorithm, the calculation of the text-similarity with the smith-waterman algorithm can be applied based on the word order [16]–[18].

The string-matching process between two strings will produce identical/similar alignment (hit) with or without string sequence changes such as deletion, insertion, and replacement [16]. The string matching between two strings $A = a_1, a_2, a_3, \dots, a_n$ and $B = b_1, b_2, b_3, \dots, b_n$ can be applied by the following steps. First, create a substitution matrix using equation (5).

$$s(a_i, b_j) = \sum_{\mathrm{i,j}}^{\mathrm{k}} s(a_i, b_j) \tag{5}$$

If there is a match, assign +1, if there is a mismatch, assign -1, and if there is a gap, assign -2 as shown in (6).

$$s(a_i, b_j) = \begin{cases} +1, a_i = b_j \\ -1, a_i \neq b_j \end{cases} \tag{6}$$

Second, create scoring matrix $M[i][j]$ where $M[i][0] = 0$ and $M[0][j] = 0$. The size of the score matrix is set to be $(length(A) + 1) * (length(B) + 1)$. Third, score each element of the scoring matrix using (7):

$$M[i][j] = max \begin{cases} M[i-1][j-1] + S(a_i, b_j) \\ M[i-1][j] + c\ if(a_i, -) \\ M[i][j-1] + c\ if(-, b_j) \end{cases} \tag{7}$$

With the input in the matrix being the best similarity value in the prefix of the two strings, c is the cost of a gap expressed in a linear penalty gap as $C_k = kC_l$ where k is the length of the gap. Fourth, traceback from the element with the highest score to an element with score 0, from the bottom to the top.

RESULT AND DISCUSSION

The results of this study are divided into two parts namely data from questionnaires and data from tweets. The results were obtained after experimenting with five repetitions.

Result of the questionnaire data

Table 5 below shows the experimental result of normalization on questionnaire data using Levenshtein Distance, Jaro-Winkler Distance, and Smith-Waterman.

Table 5. Result of the questionnaire data

Algorithm	Time(second)	Accuracy (%)
Levenshtein Distance	47.36	44.90
Jaro-Winkler Distance	31.01	33.62
Smith-Waterman	437.72	37.31

The Jaro-Winkler algorithm spent 31.01 seconds; the fastest running time compared to two other algorithms. The Levenshtein Distance algorithm spent 47.36 seconds, followed by the Smith-Waterman algorithm which spent the longest time processing the questionnaire data, 437.72 seconds. In the term of accuracy, the Levenshtein Distance algorithm scored the highest accuracy, achieving 44.90%. The Smith-Waterman and Jaro-Winkler algorithm was followed, scoring 37.31% and 33.62%, respectively. Regarding running time and accuracy score, the Levenshtein Distance algorithm is better than the two others. Although spending longer running time than the Jaro-Winkler Distance algorithm, there is no significant difference, as the difference is only a few seconds. On the other side, even when the Smith-Waterman algorithm scored higher accuracy than the Jaro-Winkler algorithm, it does not appear to be a better option, as it spent about 14 times longer training times than the other two algorithms.

Result of the tweet data
Table 6 below shows the experimental result of normalization on the tweet data using Levenshtein Distance, Jaro-Winkler Distance, and Smith-Waterman.

Table 6. Result of the tweet data

Algorithm	Time(second)	Accuracy (%)
Levenshtein Distance	107.24	60.04
Jaro-Winkler Distance	69.27	54.69
Smith-Waterman	973.27	46.77

The Jaro-Winkler Distance algorithm spent the fastest running time of 69.27 seconds, followed by the Levenshtein Distance algorithm and the Smith-Waterman algorithm, which scored 107.24 seconds and 973.27 seconds, respectively. In terms of accuracy, the Levenshtein Distance algorithm achieved 60.04%. The Jaro-Winkler Distance achieved 54.69%, and the Smith-Waterman algorithm achieved 46.77%. In line with the result of the questionnaire data, the Levenshtein Distance performs better compared to the others. Although, there are differences in the gap of time and accuracy between the Levenshtein Distance and Jaro-Winkler Distance compared to the previous section. Firstly, the gap in time between the Levenshtein Distance and Jaro-Winkler Distance is increased by around 38 seconds (around 17 seconds on the questionnaire dataset). Secondly, the gap in accuracy scores between these two algorithms is narrowed, having approximately 5.35% in differences (around 11.28% on the questionnaire dataset). Also, in this experiment, the Smith-Waterman algorithm did not score any better than others in terms of time and accuracy.

CONCLUSION
Based on the experimental results, it can be concluded that the Jaro-Winkler Distance algorithm can normalize the two Indonesian datasets quickly. However, in terms of accuracy, the Levenshtein Distance algorithm provides relatively better results compared to Jaro-Winkler and Smith-Waterman. The Smith-Waterman algorithm may not be a better choice because it consumes much time and has mediocre accuracy scores.

REFERENCES

[1] Y. Chen, Z. Ding, Q. Zheng, Y. Qin, R. Huang, and N. Shah, "A history and theory of textual event detection and recognition," *IEEE Access*, vol. 8, pp. 201371–201392, 2020.

[2] Y. Li, A. Algarni, M. Albathan, Y. Shen, and M. A. Bijaksana, "Relevance feature discovery for text mining," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 6, pp. 1656–1669, 2015.

[3] I. E. Agbehadji, H. Yang, S. Fong, and R. Millham, "The Comparative Analysis of Smith-Waterman Algorithm with Jaro-Winkler Algorithm for the Detection of Duplicate Health Related Records," *2018 Int. Conf. Adv. Big Data, Comput. Data Commun. Syst. icABCD 2018*, pp. 1–10, 2018.

[4] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Comput. Speech Lang.*, vol. 15, no. 3, pp. 287–333, 2001.

[5] G. Slamova and M. Mukhanova, "Text normalization and spelling correction in Kazakh language," *CEUR Workshop Proc.*, vol. 2268, pp. 221–228, 2018.

[6] I. G. B. B. Nugraha and R. D. Rizqullah, "Normalisasi Kata Tidak Baku yang Tidak Disingkat dengan Jarak Perubahan," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 8, no. 3, p. 218, 2019.

[7] E. Lefever, S. Labat, and P. Singh, "Identifying cognates in English-Dutch and French-Dutch by means of orthographic information and cross-lingual word embeddings," *Lr. 2020 - 12th Int. Conf. Lang. Resour. Eval. Conf. Proc.*, no. May, pp. 4096–4101, 2020.

[8] Y. Abdellah, A. S. Lhoussain, G. Hicham, and N. Mohamed, "Spelling correction for the Arabic language-space deletion errors," *Procedia Comput. Sci.*, vol. 177, pp. 568–574, 2020.

[9] K. H. Wai, Y. K. Thu, H. A. Thant, S. Z. Moe, and T. Supnithi, "String Similarity Measures for Myanmar Language (Burmese)," *Proc. First Int. Work. NLP Solut. Under Resour. Lang. (NSURL 2019) co-located with ICNLSP 2019 - Short Pap.*, pp. 94–102, 2019, [Online]. Available: https://www.aclweb.org/anthology/2019.nsurl-1.14

[10] A. Y. W. Finansyah, "Analisis Perbandingan Kinerja Algoritma Similarity Measure sebagai Tahapan Data Preprocessing: Text Normalization *Bahasa Indonesia* Untuk Analisa Sentimen," Universitas Gadjah Mada, 2020.

[11] N. Aliyah Salsabila, Y. Ardhito Winatmoko, A. Akbar Septiandri, and A. Jamal, “Colloquial Indonesian Lexicon,” *Proc. 2018 Int. Conf. Asian Lang. Process. IALP 2018*, pp. 226–229, 2019.

[12] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Sov. Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[13] Y. W. Berger, B., Waterman, M. S., & Yu, “Levenshtein distance, sequence comparison and biological database search,” *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 3287–3294, 2021.

[14] B. Kilic and F. Gülgen, “Investigating the quality of reverse geocoding services using text similarity techniques and logistic regression analysis,” *Cartogr. Geogr. Inf. Sci.*, vol. 47, no. 4, pp. 336–349, 2020.

[15] A. Yagahara, M. Uesugi, and H. Yokoi, “Identification of synonyms using definition similarities in japanese medical device adverse event terminology,” *Appl. Sci.*, vol. 11, no. 8, p. 3659, 2021.

[16] S. Y. Yuliani, S. Y. Yuliani, S. Sahib, M. F. Abdollah, Y. S. Wijaya, and N. H. M. Yusoff, “Hoax news validation using similarity algorithms,” *J. Phys. Conf. Ser.*, vol. 1524, no. 1, 2020.

[17] F. Habibie, Afiahayati, G. B. Herwanto, S. Hartati, and A. Z. K. Frisky, “A Parallel ClustalW Algorithm on Multi-Raspberry Pis for Multiple Sequence Alignment,” *Proc. - 2018 1st Int. Conf. Bioinformatics, Biotechnol. Biomed. Eng. BioMIC 2018*, no. 1, pp. 1–6, 2019.

[18] Afiahayati and S. Hartati, “Multiple sequence alignment using Hidden Markov model with augmented set based on BLOSUM 80 and its influence on phylogenetic accuracy,” *2010 Int. Conf. Distrib. Fram. Multimed. Appl.*, pp. 1–8, 2010.