Jurnal Mantik, 7 (3) (2023), ISSN 2685-4236 (Online)

Published by: Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage: www.iocscience.org/ejournal/index.php/mantik

Optimizing nazief adriani's stemmer algorithm in detecting indonesian word errors using sastrawi

Muhamad Rosidin[1], Muhammad Fauzan Gustafi[2], Septia Ayu Pratiwi[3]
[1,2,3]Fakultas Teknologi Dan Ilmu Kesehatan/Sistem Informasi, Universitas Siber Muhammadiyah, Indonesia

A R T I C L E I N F O

Article history:

Received Aug 30, 2023
Revised Sep 08, 2023
Accepted Oct 17, 2023

Keywords:

Indonesian language;
Nazief Adriani's algorithm;
Sastrawi;
Stemming;
Word errors.

ABSTRACT

The study aimed to enhance the performance of Nazief Adriani's Stemmer Algorithm for detecting word errors in Indonesian texts by utilizing the Sastrawi stemmer. The evaluation utilized a dataset containing Indonesian sentences from a specific source, consisting of 27,704 words. The processing time comparison between the two stemmer algorithms, Nazief Adriani and Sastrawi, is presented in the results. Each algorithm underwent three trials, and it was evident that Sastrawi demonstrated significantly faster processing times, averaging around 2.3254 seconds, while Nazief Adriani averaged approximately 301.7509 seconds. Additionally, the word error count detected by each algorithm in their respective trials is also presented. Nazief Adriani detected 3048 word errors in each trial, whereas Sastrawi detected an average of 1629 word errors. To provide a visual representation of the word error detection comparison, a graphical representation highlighted Sastrawi's superior performance in efficiently identifying word errors in Indonesian texts

Corresponding Author:

Muhamad Rosidin,
Fakultas Teknologi Dan Ilmu Kesehatan/Sistem Informasi,
Universitas Siber Muhammadiyah,
Jalan HOS Cokroaminoto No. 17, RT 53 RW 12 Kota Yogyakarta, 55253, Indonesia.
Email: rosidin@sibermu.ac.id

1. INTRODUCTION

Affixes are abundant in the Indonesian language, making them easy to identify as they can be applied to all words and combined in various ways (Li et al., 2021). There are three types of affixes in Indonesian: prefixes, insertions, and suffixes (Ni'mah & Syuhada, 2022). However, determining the base form of words that contain affixes is not a simple task, as some base words undergo letter changes when affixes are added, leading to difficulties in using words correctly. To address this, a stemming algorithm can be employed to transform words containing affixes into their base form.

The implementation of stemming is of utmost importance in text mining, playing a critical role in the process. For example, stemming has been effectively utilized in plagiarism research. Plagiarism is a long-standing issue in the Indonesian education system, affecting educational institutions due to the presence of plagiarized content

(Sadanand et al., 2022). In light of this, the researchers aim to develop an Indonesian word error detection system using the Nazief Adriani algorithm (SIMANJUNTAK et al., 2021).

Typing can be performed in two ways: looking at the keyboard and touch typing without looking. Currently, manually detecting word errors in a document is a challenging task (Islam et al., 2018). In the previous study by Yudhana, A., Fadlil, A., & Rosidin, M. (2019) titled "Indonesian Words Error Detection System using Nazief Adriani Stemmer Algorithm," the implementation of the stemming algorithm was successfully applied to detect word errors in the Indonesian language (Yudhana et al., 2019). The process of stemming can be carried out using various algorithms, including Nazief and Adriani, Porter, Confix Stripping, Enhanced Confix Stripping, Porter Stemmer, and Modified Porter Stemmer (Budi & Suryono, 2023).

This study addresses gaps in prior research by enhancing the stemmer algorithm using Sastrawi, improving performance parameters such as processing time and detection capability. The Sastrawi Stemmer has demonstrated superior performance in handling Indonesian texts, particularly in terms of speed and accuracy (Rosid et al., 2020). By integrating these two algorithms, the overall performance of the word error detection system is expected to be enhanced, enabling better detection of errors caused by affixes and other linguistic variations in Indonesian words (Nova Sabrina , Buku Rina, 2020). The Sastrawi algorithm is expected to handle affix handling processes and can be utilized in the development of natural language processing systems, such as plagiarism detection.

2. RESEARCH METHOD

The primary objective of this study is to enhance the performance of the Nazief Adriani Stemmer Algorithm in detecting Indonesian word errors by incorporating the Sastrawi Stemmer. The research encompasses the development of an Indonesian word error detection system, utilizing both the Nazief Adriani and Sastrawi Stemmer algorithms, The implementation process is customized to meet the specific needs of creating an information system, utilizing the Hypertext Preprocessor (PHP) programming language as the platform for development (Rosidin et al., 2022). The system is subsequently tested to ensure the accurate functioning of word error detection and to measure the frequency of word errors as well as the processing time.

The flowchart presented in Figure 1 outlines the system's process flow developed in this research. The flowchart illustrates the sequence of steps for word error detection in Indonesian text documents using either the Nazief Adriani stemmer or the Sastrawi stemmer. The system performs the word error detection process using one of the two stemmer algorithms and produces an output consisting of a list of detected erroneous words. The system's workflow concludes once the detection process is completed, and the final output is generated.

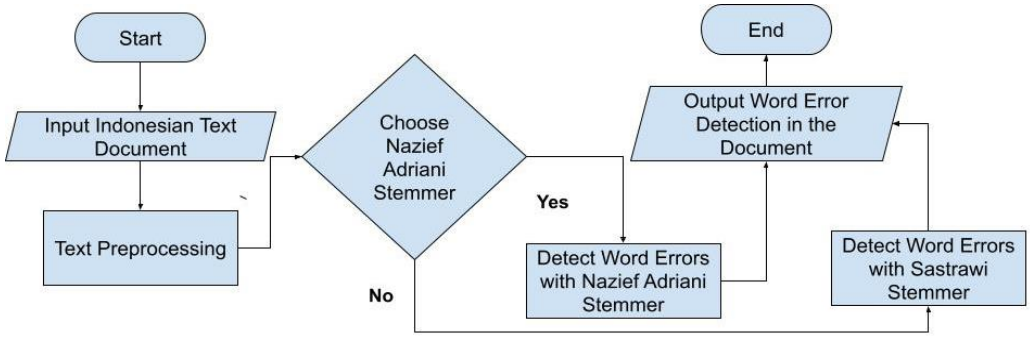

Figure 1. Flowchart of the word error detection system

Figure 1 illustrates a word error detection system for Indonesian text documents using two different stemmer methods: the Nazief Adriani stemmer and the Sastrawi stemmer. The process begins with inputting an Indonesian text document, followed by text preprocessing to ensure uniformity in analysis through actions like removing special characters and converting the text to lowercase.

Next, the system selects either the Nazief Adriani or Sastrawi stemmer method. If the Nazief Adriani stemmer is chosen (Yes), the system proceeds with word error detection using this specific stemmer. The result is an output of identified word errors that can aid in enhancing the text's quality.

If, however, the Sastrawi stemmer is chosen (No), the system directly employs this stemmer for word error detection, resulting in a list of word errors in the text document.

Upon completing the word error detection using the selected stemmer method, the system provides an output consisting of the list of word errors. The flowchart concludes once all the steps have been carried out, and the system's output can be utilized to rectify and improve the quality of Indonesian text. This flowchart, represented by Figure 1, provides valuable insights into the functioning of the word error detection system using two distinct stemmer methods, offering effective solutions to enhance the accuracy and consistency of text analysis.

Text preprocessing is a crucial step in constructing the text corpus, which involves collecting and preprocessing the data (Isa et al., 2019). It serves as an initial stage for both semantic analysis, ensuring accuracy in meaning, and syntactic analysis, ensuring accuracy in arrangement (Mantik et al., 2021). The Indonesian text processing includes various steps such as case folding, tokenizing, stopword removal, and stemming. Before stemming is performed, the document must undergo preprocessing, which consists of tokenization, case folding, filtering, and stemming (Prihatini et al., 2019).

Preprocessing aims to remove irrelevant characters and words from the document. This omission streamlines and improves word processing (Razmi et al., 2021). In text mining, preprocessing is expected to enhance processing time by eliminating unnecessary words or text from the documents (Javed & Kamal, 2018). In this study, four commonly used preprocessing methods are combined, including case folding, tokenizing, stopword removal, and stemming. (a) Case folding: This process involves converting all capital letters in a document to lowercase (a-z). In this study, case folding is performed through a function in the PHP programming language (Resyanto et al., 2019). (b) Tokenizing: Tokenizing is used to separate or eliminate input strings based on each word within the document. It involves breaking down the document into individual words, excluding numbers, characters, symbols, and punctuation in addition to the alphabet (Mantik et al., 2022). (c) Filtering: Words listed in the stopword or stoplist are removed during this step. Stopwords are commonly occurring words in large amounts of text and are considered insignificant for analysis (Khomsah & Aribowo, 2017). In this study, no words are deleted, as each word will be verified in the database. (d) Stemming: Stemming in Indonesian is more complex than in English due to the variations of affixes that need to be removed to obtain the base word (Rosidin et al., 2019). Indonesian morphology presents a higher level of complexity compared to English (Virmani & Taneja, 2019). Besides Indonesian and English, stemming can also be applied to Arabic, as seen in other research (Omar et al., 2016). Stemming involves determining the base words of words containing affixes. One of the widely used stemming algorithms is Nazief Adriani, while the Sastrawi algorithm is also effective in stemming processes. Stemming is implemented based on the appropriate affix. In Indonesian, the same intonation can result in different meanings depending on the word's topic domain. For instance, the Indonesian word "kemeja" with the same intonation can be written as "ke meja" (go to the table) or "kemeja" (a dress) (Hidayatullah & Suyanto, 2019).

The algorithm used for stemming in this research is derived from the Nazief and Adriani algorithm (Ibrohim & Budi, 2019). The flow chart diagram of the Nazief and Adriani algorithm is depicted in Figure 2.

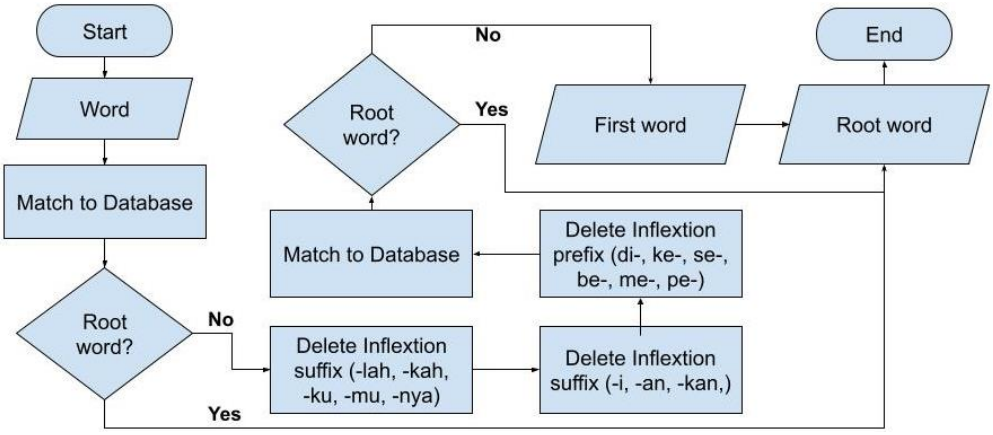

Figure 2. Flowchart of the Nazief and Adriani stemming algorithm

The flowchart in Figure 2 illustrates the process of identifying the root word using the Nazief-Adriani stemming algorithm (Rizki et al., 2019). The process begins with inputting the initial word that needs to be analyzed. First, the word is checked to see if it is already a root word or not. If the word is already a root word, the flowchart concludes. However, if it is not a root word, a series of steps are undertaken to remove affixes.

The first step involves removing suffixes (-lah, -kah, -ku, -mu, -nya). Next, the second step is to remove other suffixes (-i, -an, -kan). Lastly, the third step is to remove prefixes (di-, ke-, se-, be-, me-, pe-).

After all the affixes are removed, the remaining word is re-checked to see if it is now a root word or not. If the word has become a root word, the process ends. However, if it is still not a root word, the steps of removing affixes can be repeated if necessary.

The process continues for the first word or any new word entered as input. The flowchart will keep running until the entered word is finally identified as a root word. At that point, the flowchart ends.

Sastrawi is essentially a stemmer library, accessible on a source code provider's website through the link https://github.com/sastrawi/sastrawi. According to the review by (Mustikasari et al., 2021), this library is built upon research from. The website mentions that the stemming process using this stemmer heavily relies on the root word dictionary, which is primarily derived from kateglo.com with minor modifications. The rules of the Sastrawi stemmer can be summarized as follows: (a) First, it checks whether the word to be stemmed exists in the dictionary of root words. If it does, the process stops at this step. (b) If the word is not found in the dictionary, it implies that it contains affixes, and these suffixes (-lah, -kah, -ku, -mu, -nya, -lah, -kah, -tah, or -pun) are removed. (c) The derivative affixes -i, -kan, -an are then removed, followed by the elimination of the prefixes be-, di-, ke-, me-, pe-, se-, and te-. (d) If the root word resulting from the previous steps is still not found in the dictionary, the algorithm checks whether the word is listed in the ambiguous table in the last column or not. (e) If all the above steps fail, the algorithm eventually returns the word to its original form.

3. RESULTS AND DISCUSSIONS

In summary, this study aimed to improve the performance of Nazief Adriani's Stemmer Algorithm in detecting word errors in Indonesian texts by utilizing the Sastrawi stemmer. The evaluation involved testing a dataset containing Indonesian sentences, totaling

27,704 words. The results presented in Table 1 clearly demonstrated a substantial difference in processing times between the "Nazief Adriani" and "Sastrawi" algorithms. The "Sastrawi" algorithm showed remarkable efficiency with an average processing time of around 2.3254 seconds, while the "Nazief Adriani" algorithm had an average processing time of approximately 301.7509 seconds. These findings indicate that the "Sastrawi" algorithm is significantly more efficient in performing the stemming process for word error detection in the Indonesian language.

Furthermore, Table 2 provided insights into the word error counts detected by each algorithm during their respective trials. The "Nazief Adriani" algorithm consistently detected 3048 word errors in each trial, resulting in an average of 3048 word errors. Conversely, the "Sastrawi" algorithm detected an average of 1629 word errors in its three trials, substantially fewer than the "Nazief Adriani" algorithm.

To visually represent the comparison in word error detection, Figure 7 graphically displayed the performance of the "Nazief Adriani" and "Sastrawi" algorithms. The graph clearly illustrated the superiority of "Sastrawi" in efficiently identifying word errors in Indonesian texts, reinforcing the conclusions drawn from the test results.

In conclusion, the incorporation of the Sastrawi stemmer has proven to enhance the word error detection process, significantly reducing processing times, and improving the accuracy of error identification in Indonesian texts. These results underscore the importance of using efficient algorithms for stemming in natural language processing tasks, optimizing text analysis, and error detection capabilities.

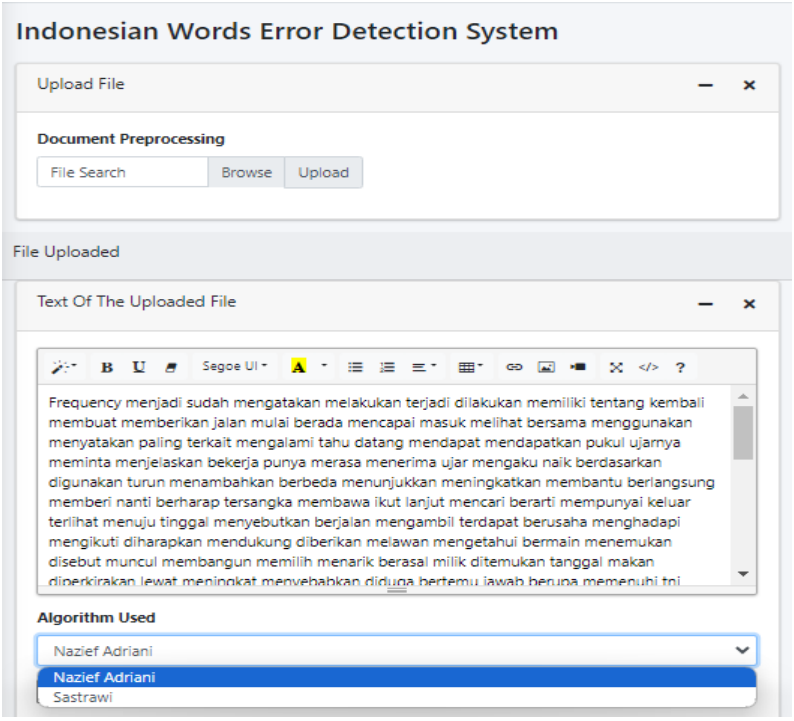

Figure 3. Interface of Indonesian Words Error Detection System.

Figure 4. Interface of Indonesian Words Error Detection System.

Furthermore, apart from the page featuring word error highlighting, as shown in Figure 5, users are provided with a table displaying the results of the stemming process. They have the option to select either Sastrawi or Nazief and Adriani's algorithm for stemming. The table presents words containing affixes, along with their respective stemming outcomes. However, for the words identified as errors, the stemming results remain hidden. The table displaying the stemming results can be found in Figure 5.

Kata Asli	Hasil Stemming
Frequency	
menjadi	menjadi
sudah	sudah
mengatakan	mengatakan
melakukan	melakukan
terjadi	terjadi
dilakukan	dilakukan
memiliki	memiliki
tentang	tentang
kembali	kembali
membuat	membuat
memberikan	memberikan
jalan	jalan

Figure 5. The Interface of Stemming Result Table.

The documents intended for testing in the word error detection system consist of Indonesian sentences. These documents have been sourced from a dataset accessible on the website: https://figshare.com/articles/dataset/Dataset/7857050?file=14629016. The total word count in these documents amounts to 27,704 words.

Table 1 displays the processing time results for the "Nazief Adriani" and "Sastrawi" stemmer algorithms during testing. Each row corresponds to a different trial.

In the "Nazief Adriani" algorithm, three trials were conducted. Trial 1 took approximately 306.522 seconds, Trial 2 was slightly faster at around 298.6182 seconds, and Trial 3 reached about 299.1115 seconds. The average processing time for the "Nazief Adriani" algorithm was approximately 301.7509 seconds.

On the other hand, the "Sastrawi" algorithm also underwent three trials. Trial 1 had a processing time of only about 2.3243 seconds, Trial 2 was slightly shorter at approximately 2.3069 seconds, and Trial 3 took around 2.3451 seconds. The average processing time for the "Sastrawi" algorithm was approximately 2.3254 seconds.

The test results clearly indicate that the "Sastrawi" algorithm significantly outperforms the "Nazief Adriani" algorithm in terms of processing time. This implies that "Sastrawi" is more efficient in performing the stemming process to detect word errors in the Indonesian language.

Table 1. Processing Time Comparison of Nazief Adriani and Sastrawi Stemmer Algorithms

Stemmer	PROCESSING TIME (seconds)
Nazief Adriani Trial 1	306.522
Nazief Adriani Trial 2	298.6182
Nazief Adriani Trial 3	299.1115
Sastrawi Trial 1	2.3243
Sastrawi Trial 2	2.3069
Sastrawi Trial 3	2.3451

The information described earlier is visually depicted in Figure 6, which serves as a graphical representation of the processing time comparison between the "Nazief Adriani" and "Sastrawi" stemmer algorithms.

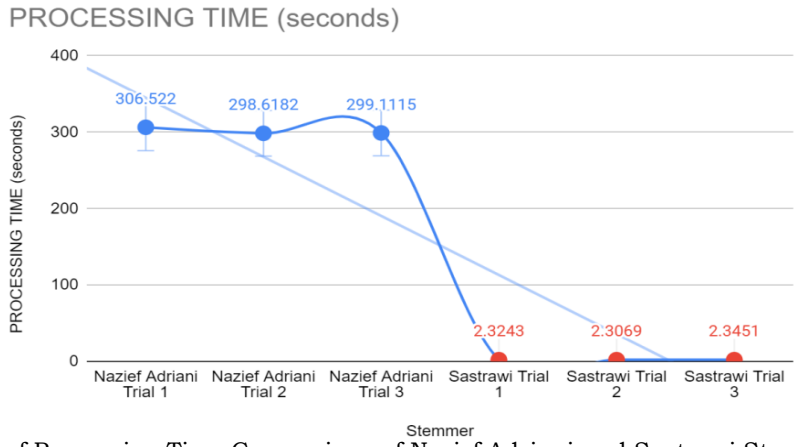

Figure 6. Chart of Processing Time Comparison of Nazief Adriani and Sastrawi Stemmer Algorithms

Figure 6 illustrates the comparison of processing time between Nazief Adriani and Sastrawi stemmer algorithms. Sastrawi stemmer shows significantly shorter processing time (2 to 2.35 seconds) compared to Nazief Adriani stemmer (298 to 306 seconds), indicating that Sastrawi stemmer is more efficient in text processing.

Table 2 shows the word error count (ERROR) detected by the "Nazief Adriani" and "Sastrawi" stemmer algorithms. Each algorithm underwent three trials, with the "Nazief Adriani" detecting 3048 word errors in each trial, resulting in an average of 3048 errors. Likewise, the "Sastrawi" algorithm detected 1629 word errors in each of its three trials, with an average of 1629 errors.

Table 2. Word error detection Comparison of Nazief Adriani and Sastrawi Stemmer Algorithms

Stemmer	Word Error Detection (word)
Nazief Adriani Trial 1	3048
Nazief Adriani Trial 2	3048
Nazief Adriani Trial 3	3048
Sastrawi Trial 1	1629
Sastrawi Trial 2	1629
Sastrawi Trial 3	1629

To visually represent this information, a graphical depiction will be presented in Figure 7. Figure 7 will illustrate the comparison of word errors detected by the "Nazief Adriani" and "Sastrawi" algorithms. Each data point on the x-axis represents a trial (Trial 1, Trial 2, Trial 3), while the y-axis represents the count of word errors (ERROR).

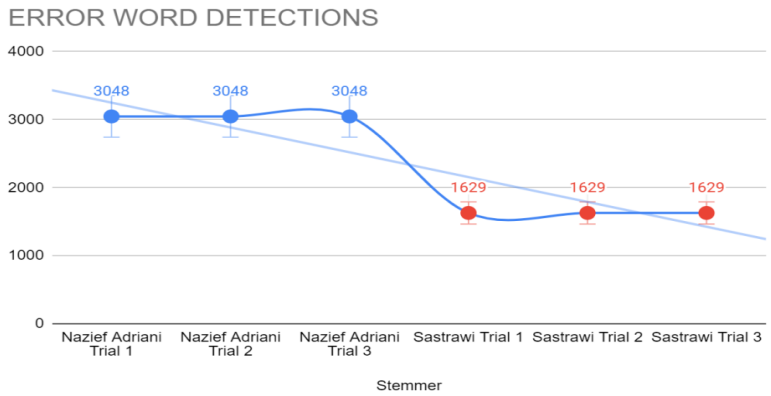

Figure 7. Chart of Word error detection Comparison of Nazief Adriani and Sastrawi Stemmer Algorithms

The graph will provide a clear visualization of the difference in word error detection between the two algorithms, emphasizing the superior performance of "Sastrawi" in identifying word errors in the Indonesian language.

The incorporation of the Sastrawi stemmer in the word error detection process has proven to bring significant enhancements in both processing times and accuracy. The faster processing times and lower number of word errors detected by the "Sastrawi" algorithm emphasize its effectiveness in optimizing text analysis and error detection capabilities for Indonesian language processing tasks. These findings highlight the importance of utilizing efficient algorithms like the "Sastrawi" stemmer in natural language processing applications, especially for languages with intricate word structures like Indonesian. The improved performance of the "Sastrawi" algorithm can contribute to more precise and efficient text analysis, benefiting various language-related applications and research domains.

4. CONCLUSION

In summary, this study aimed to improve the performance of Nazief Adriani's Stemmer Algorithm in detecting word errors in Indonesian texts by utilizing the Sastrawi stemmer. The evaluation involved testing a dataset containing Indonesian sentences, totaling 27,704 words. The results presented in Table 1 clearly demonstrated a substantial difference in processing times between the "Nazief Adriani" and "Sastrawi" algorithms. The "Sastrawi" algorithm showed remarkable efficiency with an average processing time of around 2.3254 seconds, while the "Nazief Adriani" algorithm had an average processing time of approximately 301.7509 seconds. These findings indicate that the "Sastrawi" algorithm is significantly more efficient in performing the stemming process for word error detection in the Indonesian language.

Furthermore, Table 2 provided insights into the word error counts detected by each algorithm during their respective trials. The "Nazief Adriani" algorithm consistently detected 3048 word errors in each trial, resulting in an average of 3048 word errors. Conversely, the "Sastrawi" algorithm detected an average of 1629 word errors in its three trials, substantially fewer than the "Nazief Adriani" algorithm.

In conclusion, the incorporation of the Sastrawi stemmer has proven to enhance the word error detection process, significantly reducing processing times, and improving the accuracy of error identification in Indonesian texts. These results underscore the importance of using efficient algorithms for stemming in natural language processing tasks, optimizing text analysis, and error detection capabilities.

This research shows that using the Sastrawi stemmer to detect word errors in Indonesian texts has enhanced efficiency and accuracy in error identification, potentially benefiting text analysis and natural language processing. However, the study's contributions lie in developing a more efficient stemming algorithm and providing empirical evidence on performance differences between the "Nazief Adriani" and "Sastrawi" algorithms in the context of error detection. Nonetheless, the study's limitations include its focus on a specific language and dataset, necessitating further research to generalize these findings across various languages and text processing contexts.

REFERENCES

Budi, I., & Suryono, R. R. (2023). Application of named entity recognition method for Indonesian datasets: a review. *Bulletin of Electrical Engineering and Informatics*, *12*(2), 969–978. https://doi.org/10.11591/eei.v12i2.4529

Hidayatullah, S. N., & Suyanto. (2019). Developing an adaptive language model for Bahasa Indonesia. *International Journal of Advanced Computer Science and Applications*, *10*(1), 488–492. https://doi.org/10.14569/IJACSA.2019.0100163

Ibrohim, M. O., & Budi, I. (2019). *Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter*. 46–57. https://doi.org/10.18653/v1/w19-3506

Isa, S. M., Suwandi, R., & Andrean, Y. P. (2019). Optimizing the hyperparameter of feature extraction and machine learning classification algorithms. *International Journal of Advanced Computer Science and Applications*, *10*(3), 69–76. https://doi.org/10.14569/IJACSA.2019.0100309

Islam, M. I. K., Habib, M. T., Rahman, M. S., Rahman, M. R., & Ahmed, F. (2018). A context-sensitive approach to find optimum language model for automatic Bangla spelling correction. *International Journal of Advanced Computer Science and Applications*, *9*(11), 184–191. https://doi.org/10.14569/ijacsa.2018.091126

Javed, M., & Kamal, S. (2018). Normalization of unstructured and informal text in sentiment analysis. *International Journal of Advanced Computer Science and Applications*, *9*(10), 78–85. https://doi.org/10.14569/IJACSA.2018.091011

Khomsah, S., & Aribowo, A. S. (2017). Text-Preprocessing Komentar Youtube Dalam Bahasa Indonesia. *Rekayasa Sistem Dan Teknologi Informasi* , *1*(3), 648–654.

Li, Y., Du, T., Zhu, L., & Qu, S. (2021). An Efficient Minimal Text Segmentation Method for URL Domain Names. *Scientific Programming*, *2021*, 9946729. https://doi.org/10.1155/2021/9946729

Mantik, J., Indra, S., & Situmeang, G. (2021). 2022) 423-430 Accredited. *Jurnal Mantik*, *6*(1), 423–430.

Mantik, J., Saputri, Y. R., & Februariyanti, H. (2022). Sentiment Analysis on Shopee E-Commerce Using the Naïve Bayes Classifier Algorithm. *Jurnal Mantik*, *6*(2), 1349–1357.

Mustikasari, D., Widaningrum, I., Arifin, R., & Putri, W. H. E. (2021). Comparison of Effectiveness of Stemming Algorithms in Indonesian Documents. *Proceedings of the 2nd Borobudur International Symposium on Science and Technology (BIS-STE 2020)*, *203*, 154–158. https://doi.org/10.2991/aer.k.210810.025

Ni'mah, A. T., & Syuhada, F. (2022). Term Weighting Based Indexing Class and Indexing Short Document for Indonesian Thesis Title Classification. *Journal of Computer Science and Informatics Engineering (J-Cosine)*, *6*(2), 167–175.

Nova Sabrina , Buku Rina, C. S. (2020). Jurnal Mantik Jurnal Mantik. *Mobile-Based National University Online Library Application Design*, *3*(2), 10–19. http://iocscience.org/ejournal/index.php/mantik/article/view/882/595

Omar, H., Dahab, M., & Kamal, M. (2016). Stemmer Impact on Quranic Mobile Information Retrieval Performance. *International Journal of Advanced Computer Science and Applications*, *7*(12), 135–139. https://doi.org/10.14569/ijacsa.2016.071218

Prihatini, P. M., Putra, I. K. G. D., Giriantari, I. A. D., & Sudarma, M. (2019). Complete agglomerative hierarchy document's clustering based on fuzzy Luhn's gibbs latent dirichlet allocation. *International Journal of Electrical and Computer Engineering*, *9*(3), 2103–2111. https://doi.org/10.11591/ijece.v9i3.pp2103-2111

Razmi, N. A., Zamri, M. Z., Ghazalli, S. S. S., & Seman, N. (2021). Visualizing stemming techniques on online news articles text analytics. *Bulletin of Electrical Engineering and Informatics*, *10*(1), 365–365. https://doi.org/10.11591/eei.v10i1.2504

Resyanto, F., Sibaroni, Y., & Romadhony, A. (2019). Choosing The Most Optimum Text Preprocessing Method for Sentiment Analysis: Case:iPhone Tweets. *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 1–5. https://doi.org/10.1109/ICIC47613.2019.8985943

Rizki, A. S., Tjahyanto, A., & Trialih, R. (2019). Comparison of stemming algorithms on Indonesian text processing. *Telkomnika (Telecommunication Computing Electronics and Control)*, *17*(1), 95–102. https://doi.org/10.12928/TELKOMNIKA.v17i1.10183

Rosid, M. A., Fitrani, A. S., Astutik, I. R. I., Mulloh, N. I., & Gozali, H. A. (2020). Improving Text Preprocessing for Student Complaint Document Classification Using Sastrawi. *IOP Conference Series: Materials Science and Engineering*, *874*(1). https://doi.org/10.1088/1757-899X/874/1/012017

Rosidin, M., Fadlil, A., & Yudhana, A. (2019). Sistem Kelas Kata Berimbuhan Menggunakan Algoritma Porter Stemmer Sebagai Pembelajaran Bahasa Indonesia. *Telematika*, *16*(1), 11–17.

Rosidin, M., Sulistyo, W. Y., Setyaputri, K. E., & Supriyanto, J. (2022). Rancang Bangun Sistem Informasi Pendataan Beasiswa PTMA Berbasis Web Menggunakan Metode Waterfall. *Jurnal Riset Teknologi Informasi Dan Komputer (JURISTIK)*, *2*(1), 7–11. https://doi.org/10.53863/juristik.v2i1.474

Sadanand, V. S., Guruvyas, K. R. R., Patil, P. P., Acharya, J. J., & Suryakanth, S. G. (2022). An automated essay evaluation system using natural language processing and sentiment analysis. *International Journal of Electrical and Computer Engineering*, *12*(6), 6585–6593. https://doi.org/10.11591/ijece.v12i6.pp6585-6593

SIMANJUNTAK, M., Panjaitan, J., & Syahputra, S. A. (2021). Using Preprocessing Text Mining With Nazief-Adriani Algorithms Similarity Of Essay Final Exam Semester. *Jurnal Mantik*, *5*(36), 1714–1720. http://www.iocscience.org/ejournal/index.php/mantik/article/view/1544%0Ahttp://www.iocscience.org/ejournal/index.php/mantik/article/download/1544/1329

Virmani, D., & Taneja, S. (2019). *A Text Preprocessing Approach for Efficacious Information Retrieval BT - Smart Innovations in Communication and Computational Sciences* (B. K. Panigrahi, M. C. Trivedi, K. K. Mishra, S. Tiwari, & P. K. Singh (eds.); pp. 13–22). Springer Singapore.

Yudhana, A., Fadlil, A., & Rosidin, M. (2019). Indonesian Words Error Detection System using Nazief Adriani Stemmer Algorithm. *IJACSA) International Journal of Advanced Computer Science and Applications*, *10*(12). https://doi.org/10.14569/IJACSA.2019.0101231