Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

Comparison of Nazief–Adriani and Porter Stemmer in Determining Javanese Root Words

Anggi Ayu Maharani, Fadhli Almu'iini Ahda*

Faculty of Technology and Design, Department of Informatics Engineering, Institut Teknologi dan Bisnis Asia Malang, Malang
Jl. Soekarno Hatta Jl. Rembuksari No.1 A, Mojolangu, Kec. Lowokwaru, Kota Malang, Jawa Timur, Indonesia
Email: [1]ayrani780@gmail.com, [2*]fadhlial@asia.ac.id,
Email Penulis Korespondensi: fadhlial@asia.ac.id
Submitted: **23/12/2025**; Accepted: **27/05/2026**; Published: **05/07/2026**

Abstrak−Stemming is an important step in text processing to convert inflected words into their root word. In Javanese, the stemming process is more challenging due to its complex morphological characteristics, including prefixes, infixes, suffixes, and combinations of affixes that are often accompanied by phonological changes. This study aims to compare the performance of the Nazief–Adriani and Porter Stemmer algorithms in Javanese stemming by including infix processing as part of the stemming stage. The dataset used consists of 603 Javanese affixed words covering various types of affixes. The evaluation was conducted using a confusion matrix with accuracy, precision, recall, and F1-score metrics, as well as an analysis of over-stemming and under-stemming errors. The results show that the Nazief–Adriani algorithm performs better with an accuracy of 90.05%, compared to the Porter Stemmer, which achieved 76.12%. This advantage is influenced by the validation of the dictionary at each stage of affix cutting, so that the stemming results are more controlled. The application of infix processing has also been proven to contribute to improving the accuracy of stemming results. This study is expected to be a reference in the development of natural language processing systems for Javanese and encourage further research related to the refinement of morphological rules.

Keywords: Stemming; Javanese Language; Nazief–Adriani; Porter Stemmer

1. INTRODUCTION

The rapid development of information technology has brought about major changes in the way humans produce, manage, and utilize information [1], [2]. One of the most noticeable impacts is the increase in the amount of digital text originating from various sources, such as social media, news portals, online forums, and digital archives in various languages, including regional languages. In the field of Natural Language Processing (NLP), the ability of systems to automatically understand and process text is very important in supporting various applications, such as information retrieval, text classification, language translation, and sentiment analysis [3], [4], [5], [6] . One important stage in text processing is stemming, which is the process of converting inflected words into their root word [7].

However, the stemming process is not always easy to implement, especially in regional languages such as Javanese. Javanese has complex morphological characteristics, marked by the use of various types of affixes, including prefixes (ater-ater), infixes (seselan), suffixes (panambang), and combinations of affixes [8], [9] . In addition, the process of affixation in Javanese is often accompanied by phonological changes, such as the softening of initial consonants, which further complicates the determination of the root word. This complexity means that many commonly used stemming algorithms cannot be applied directly without specific adjustments.

In the context of text processing, one of the main problems that still exists is the lack of an optimal stemming algorithm for Javanese [10], [11] Most stemming algorithms are developed for languages with simple morphological structures, such as English or Indonesian. When these algorithms are applied to Javanese, errors often occur in the form of over-stemming, which is excessive word truncation, and under-stemming, which is incomplete word truncation. Therefore, further study of existing stemming algorithms is needed to determine the most appropriate method for handling the characteristics of Javanese.

Two commonly used and widely discussed stemming algorithms in research are Nazief–Adriani and Porter Stemmer. The Nazief–Adriani algorithm uses a dictionary-based approach and morphological rules, so that each affix removal result is validated against the root words available in the dictionary. Meanwhile, the Porter Stemmer uses a rule-based approach, which makes the stemming process simpler and more efficient, but has the potential to produce errors in languages with complex morphological structures [12]. Although these two algorithms have been widely used, research that directly compares the two algorithms in Javanese is still limited.

Several studies have discussed the Javanese stemming process using the Nazief–Adriani algorithm. A study titled "Stemming Javanese Affix Words Using Nazief and Adriani Modifications" shows that modifications to the Nazief–Adriani algorithm for Javanese are capable of achieving a high level of accuracy in determining root words [13]. Another study titled "The Stemming Application on Affixed Javanese Words by Using Nazief and Adriani Algorithm" also proves the effectiveness of this algorithm in processing affixed Javanese words [14]. On the other hand, a study titled "Comparison of Porter's Stemming Algorithm and Nazief & Adriani" compares the two algorithms in the context of Indonesian and shows that the Nazief–Adriani algorithm has a slightly higher accuracy rate than Porter Stemmer, even though Porter excels in terms of processing speed [15]. From these various studies, a research gap can be identified, namely the absence of a study that directly compares the Nazief–Adriani and

Copyright © 2026 The Author, Page 808
This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

Porter Stemmer algorithms in the context of Javanese, taking into account the complexity of its morphological structure.

In addition, most previous studies have focused on prefix and suffix processing, while the handling of Javanese infixes has not been widely developed [16], [17]. In fact, infixes such as -in-, -um-, -el-, and -er- play an important role in Javanese word formation and can affect the accuracy of stemming results [18]. Inaccuracies in handling infixes have the potential to cause stemming errors, either in the form of over-stemming or under-stemming [14].

To complete the research, this study conducted a direct comparison between the Nazief–Adriani and Porter Stemmer algorithms in the context of Javanese, incorporating infix processing as part of the stemming stage. In this study, the Porter algorithm was adapted by adding Javanese affix cutting rules, while the Nazief–Adriani algorithm was used as a comparison with a dictionary-based and morphological rule-based approach. The evaluation was carried out using a confusion matrix with accuracy, precision, recall, and F1-score metrics, as well as an analysis of over-stemming and under-stemming errors. This research is expected to contribute to the development of science in the field of natural language processing, especially for regional languages. Practically, the results of this study are expected to be used to improve the performance of various NLP-based applications that use Javanese, such as information retrieval and text analysis systems. With a more accurate Javanese stemming algorithm that is in line with its morphological characteristics, the use of regional languages in the digital environment can be carried out more optimally and sustainably.

2. RESEARCH METHODOLOGY

This study focuses on the application and comparison of two Javanese stemming algorithms, namely Nazief–Adriani and Porter Stemmer. These two algorithms have different morphological processing methods. Nazief-Adriani is dictionary- and rule-based, while Porter is rule-based [13], [15].

Javanese has complex morphological characteristics, marked by a variety of affixes in the form of prefixes, infixes, suffixes, and combinations of affixes that are often accompanied by phonological changes such as the softening of initial consonants [19], [20]. This complexity makes the process of determining the root word more challenging than in languages with simpler morphological structures [21]. Therefore, a stemming algorithm is needed that is capable of accurately handling word formation variations so that Javanese text processing can be carried out correctly.

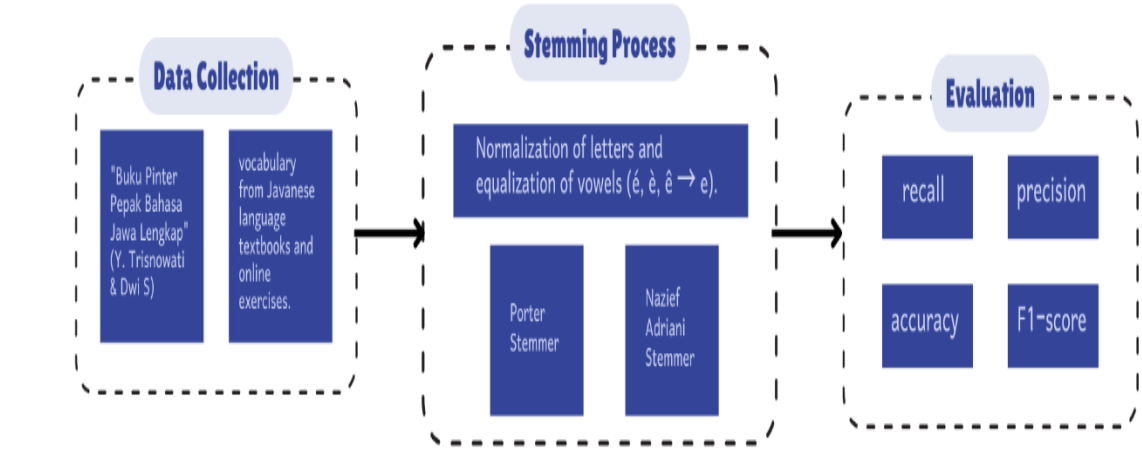

Figure 1. Research Process

Figure 1 shows that the research process begins with the data collection stage, where 717 Javanese root words are collected from Buku Pinter Pepak Bahasa Jawa Lengkap by Y. Trisnowati and Dwi S, and 603 Javanese affixed words are compiled from Javanese language textbooks and online sources. The collected data then enters the stemming process, which includes letter normalization and vowel standardization, followed by stemming of the affixed words using two algorithms, namely the Porter Stemmer and Nazief–Adriani Stemmer, to obtain their root words. The final stage is evaluation, where the stemming results from both algorithms are compared with the reference root words using a confusion matrix to calculate accuracy, precision, recall, and F1-score. This evaluation provides a basis for analyzing the differences in performance between the two algorithms in handling Javanese morphological characteristics and for determining the more suitable stemming method.

2.1 Data Collection

The data collection stage was carried out as the first step in preparing the data to be used in testing the Javanese stemming algorithm. The data used consisted of Javanese affixed words. The basic words in this study were obtained from the revised edition of Buku Pinter Pepak Bahasa Jawa Lengkap by Y. Trisnowati and Dwi S [22].

Copyright © 2026 The Author, Page 809
This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

In addition, the data was also supplemented with vocabulary from Javanese language textbooks and online exercises.

In addition to basic words, this study uses 603 affixed words compiled based on affix patterns commonly used in Javanese. The affixes used include prefixes, infixes, suffixes, and combinations of affixes. The number of affixed words based on affix type is shown in Table 1. This table provides an overview of the data distribution used in the study.

Table 1. Number of Derived Words Based on Affix Type

Type of Affix	Amount
Suffix	138
Prefix	231
Prefix + Suffix	200
Infix	20
Infix + Sufifx	14
Total	603

2.2 Porter Stemmer

Porter Stemmer is a stemming algorithm that works with a rule-based approach [15], [23]. This algorithm performs affixation based on predetermined affix patterns, so that the stemming results are entirely dependent on the rules applied [12]. In this study, Porter Stemmer performed preprocessing, which involved converting letters to lowercase and standardizing the vowels é, è, and ê to the letter e in order to facilitate the stemming process and improve the quality of the stemming results [24]. Three stages of infix, prefix, and suffix cutting. Each type of affix is only cut once, as long as the word length is not less than 3 letters. In addition, Porter Stemmer also applies softening rules to certain prefixes such as ng-, ny-, n-, and m- in accordance with common sound change patterns.

2.3 Nazief-Adriani Stemmer

The Nazief–Adriani algorithm is a dictionary-based stemming algorithm that combines morphological rules with a root word validation process [12] [13], [15]. In this study, the Nazief–Adriani algorithm was applied to process Javanese affixed words in stages. The stemming stages in the Nazief–Adriani algorithm begin with preprocessing, such as Porter. Then, infix cutting is performed, followed by prefix cutting, and finally suffix cutting. Each affix cutting result is always validated against the root word dictionary. If the cutting result is found in the dictionary, the process is stopped and the word is accepted as a root word. If it is not found, it will proceed to the next cutting stage.

2.4 Evaluation Metrics

The evaluation stage was conducted to measure the performance of both algorithms in generating the correct root words. The evaluation was carried out using a confusion matrix, which enabled the calculation of quantitative metrics such as accuracy, precision, recall, and F1-score. Precision measures the accuracy of the stemming results produced by the algorithm, while recall indicates the algorithm's ability to find all the correct root words. F1-score is used as a measure of the balance between precision and recall. Accuracy provides an overview of the proportion of correct stemming results to the entire test data. The evaluation formula used in this study is shown in Equations (1) – (4):

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (2)$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)$$

$$Accuracy = \frac{TP}{N} \times 100\% \quad (4)$$

In this study, under-stemming errors are categorized as false positives (FP), while over-stemming errors are categorized as false negatives (FN). Through this evaluation stage, a comprehensive overview of the advantages and limitations of the Nazief–Adriani and Porter Stemmer algorithms in the Javanese language stemming process is obtained.

3. RESULT AND DISCUSSION

3.1 Preprocessing Data

The preprocessing stage was carried out to standardize the data before the stemming process. The research dataset consisted of 717 root words obtained from the revised edition of Buku Pinter Pepak Bahasa Jawa Lengkap by Y.

Copyright © 2026 The Author, Page 810
This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

Trisnowati and Dwi S. Additional sources came from Javanese language textbooks and online exercises. The combination of these sources aims to enrich the coverage of base words. In addition to base words, this study uses 603 affixed words. Affixed words are arranged based on affix patterns commonly used in Javanese. Affixes include prefixes, infixes, suffixes, and combinations of affixes.

All affixes are then normalized by converting them to lowercase letters. The special vowels é, è, and ê are standardized to the letter e. For example, the word nganyaraké is normalized to nganyarake. The word ngepénakaké is also changed to ngepenakake. Normalization is performed to avoid differences in spelling during stemming processing. The stemming algorithm works based on character patterns, not pronunciation. The preprocessing stage produces consistent data that is ready to be processed by the stemming algorithm. In Javanese, the spelling and pronunciation of words often differ in everyday practice. For example, the word nyilih is often pronounced nyeleh in spoken conversation. The word nggawa is also often pronounced as nggowo. Although the spelling and pronunciation are different, the meaning remains the same.

3.2 Stemming Process

Stemming is the process of extracting root words, in which inflected words are processed through a series of affix removal rules that include infixes, prefixes, and suffixes. This study applies two stemming algorithms with different approaches, namely Nazief–Adriani (NA) and Porter Stemmer. The Nazief–Adriani algorithm is a dictionary-based method, where each affix cutting result is validated against a root word dictionary. This approach makes NA more suitable for languages with complex morphology such as Javanese. In contrast, Porter Stemmer uses a rule-based approach without dictionary validation, so the accuracy of the stemming results is highly dependent on the rules applied and has the potential to produce certain error patterns.

In its implementation, Porter Stemmer performs infix, prefix, and suffix truncation directly and only once each, while still paying attention to the minimum word length restriction. This process includes applying luluh rules to certain prefixes and removing productive Javanese suffixes without dictionary checks. Meanwhile, the Nazief–Adriani algorithm is run in stages by combining morphological rules and checking the dictionary at each stage of cutting. Each segmentation result is checked against the dictionary before proceeding to the next stage, so that the resulting base word form is more controlled. This approach makes Nazief–Adriani more adaptive in handling affix variations and phonological changes in Javanese.

3.2.1 Infix Stemming

At the infix cutting stage, both algorithms first identify the presence of infixes that are morphologically inserted after the first letter of the base word, namely in, um, el, and er. In the Nazief–Adriani algorithm, the check is performed starting from the second position of the word. If an infix is detected, the algorithm forms a candidate root word by retaining the first letter and removing the infix segment, then reconnecting the rest of the word. For certain initial patterns such as nel-, nin-, num-, ner-, mel-, min-, mum-, and mer-, the cutting result is not immediately accepted, but is first validated against the dictionary. If the resulting form is not found in the dictionary, the stemming process is canceled and the word is returned to its original form to prevent over-stemming. For words outside these patterns, the infix stemming result is accepted as a temporary form for the next stage. In contrast, Porter's algorithm applies infix cutting with a simpler and more direct approach. The process is only performed on words longer than four letters, then the algorithm checks whether the part after the first letter begins with one of the specified infixes. If detected, the infix is removed once while retaining the first letter, and the result is accepted if the length of the word still meets the minimum limit of three letters. The result obtained is immediately used in the next stage. This approach makes Porter more efficient, but it is more prone to errors if the infix rules do not fully correspond to the morphological variations of the Javanese language.

Based on the test results, variations in stemming results were found between the Nazief–Adriani and Porter algorithms, where for some infixed words such as “gumuyu,” both algorithms successfully produced the root word “guyu.” However, for other words such as “kerelip,” there was a difference in results because the Porter algorithm produced “lip”, which indicated over stemming, while the Nazief–Adriani algorithm produced the correct “kelip.” In addition, for infixed words accompanied by suffix combinations, the Nazief–Adriani algorithm was able to produce the correct root word for all test data, while the Porter algorithm still produced errors in three words, one of which was “kinalungan”, which was cut into ‘lung’ by Porter, while Nazief–Adriani produced the root form “kalung”.

Table 2. Infix Stemming Results for NA and Porter

Type	Result Category	NA Results	Porter Results
Infix	Success	19	13
	Under	0	
	Over	1	7
	Total	20	20
Infix + Suffix	Success	14	11
	Under	0	0

Copyright © 2026 The Author, Page 811
This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

Type	Result Category	NA Results	Porter Results
	Over	0	3
	Total	14	14

Based on Table 2, which shows the infix stemming results of the Nazief–Adriani and Porter algorithms, there is a clear difference in performance between the two types of words tested. For words containing only infixes, the Nazief–Adriani algorithm successfully produced the correct stemming for 19 out of 20 words, with one case of over-stemming and no cases of under-stemming. Meanwhile, Porter only produced 13 correct words, with seven cases of over-stemming. Furthermore, for words containing infixes and suffixes, the Nazief–Adriani algorithm showed consistent results with all 14 words being stemmed correctly without any errors. Meanwhile, Porter only produced 11 correct words and still experienced three cases of over stemming.

3.2.2. Prefix Stemming

In the prefix cutting stage, both algorithms identify productive initial affixes in Javanese, such as di-, ke-, pa-, sa-, ka-, ma-, ng-, ny-, n-, and m-, as well as complex prefixes such as dipun-, kapi-, kami-, pri-, pra-, and tar-. In the Nazief–Adriani algorithm, cutting is done gradually and selectively by first detecting prefixes that trigger phonological changes (liquefied), for example ny-, which can change to c- or s-, ng- to k-, and n- and m-, which affect the initial consonant of the base word. The prefix dipun- is processed with dictionary validation, and if it is not found, the stemming continues with di-. For the prefix ma-, the softening process is performed first, then the result is checked against the dictionary before the possible stemming of m- is performed. Each stemming result is validated against the dictionary, and if the remaining word is only three letters long, the process is stopped to prevent over stemming, while words with more than three letters can proceed to the suffix stemming stage.

In contrast, Porter's algorithm applies prefix truncation directly based on rules without dictionary validation. The process begins with the application of rules to the prefixes ny-, ng-, n-, and m-, including phonological changes that follow vowel characters. The prefix ma- is truncated if the length of the word meets a certain limit, then the algorithm checks the prefix, noting that if it has already been truncated, the prefix does not need to be removed again. Cutting is performed as long as the length of the cut word still meets the minimum limit of three letters, and the result is immediately accepted as a temporary form for the suffix cutting stage. This approach makes Porter simpler and more efficient, but more prone to over stemming errors in words with complex Javanese morphological structures.

Table 3. Results of Prefix Stemming Nazief – Adriani and Porter

No	Word	Affix	NA Results	Porter Result
1	dituku	di-	tuku	tuku
2	nggawe	ng-	gawe	ngawe → gaw
3	kecubles	ke-	cubles	cubles
…	…..	……	….	…..
11	dipundhut	di-	pundhut	dipundhut → dhut
12	nerusne	n-….-ne	terus	tus

Table 3 shows that both algorithms work well on simple prefixes such as “dituku” and “kecubles”. For the word “ngawe”, the correct base form is “gawe” without a layered affix structure. The Nazief–Adriani algorithm produces “gawe”, while Porter produces “gaw”. Porter error occurs because it incorrectly detects the presence of a suffix in the stemming process. For the word “dipundhut”. Nazief–Adriani produces “pundhut,” while Porter produces “dhut”. This occurs because Porter detects the prefix dipun- instead of di-. The word ‘nerusne’ in Porter is detected as having an infix, producing the word “tus”, while in NA it produces the base word “terus”.

3.2.3. Suffix Stemming

In the suffix removal stage, both stemmer handle suffixes commonly used in Javanese, such as -ake, -ke, -ipun, -nipun, -an, -ana, -en, -na, -ne, -mu, -ku, -i, -a, and -e. In the Nazief–Adriani algorithm, suffix removal is done gradually and is more selective. Certain suffixes, particularly -ake and -ke, are processed with special rules that consider the length of the resulting word so as not to damage the structure of the base word. After the suffix is removed, the resulting word form is checked again through the dictionary. If the result of the truncation is found in the dictionary, the process is complete. However, if it is not found, this algorithm still allows suffix truncation to be performed up to two times as long as the word length is greater than or equal to four and if the remaining three are returned to the final stemming result. This is to prevent over stemming.

Unlike Nazief–Adriani, the Porter algorithm applies suffix trimming using a simpler rule-based approach without checking a dictionary. After all stages of infix and prefix trimming are complete, Porter removes one suffix from the list of productive Javanese suffixes if that suffix is found at the end of the word and the remaining word length still meets the minimum limit of three letters. Suffix cutting in Porter is only done once and is not iterative, so the result of the cutting is immediately accepted as the final stemming result. This approach makes Porter more

This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

computationally efficient, but also more prone to cutting errors in words that have complex suffixes or require validation to ensure the accuracy of the base word.

Table 4. Results of suffix stemming on NA and Porter

Type	Result Category	NA Results	Porter Results
suffix	Success	121	87
	Under	0	7
	Over	17	44
Total		138	138

Table 4 above shows the stemming results of the Nazief–Adriani and Porter algorithms. It can be seen that the Nazief–Adriani algorithm produces a higher number of successful stemming, namely 121 words, compared to the Porter algorithm, which only reaches 87 words. In the Nazief–Adriani algorithm, there were no cases of under-stemming, while in the Porter algorithm, there were still seven words that experienced under-stemming. In addition, the number of over stemming in the Porter algorithm was also much greater than in Nazief–Adriani. An example of the correct results in both algorithms can be seen in the word "bapakku", where the suffix -ku was successfully removed to produce the root word "bapak". However, a difference in results can be seen in the word "ukurane", Porter only produces "ukura" , which is considered under stemming, while NA successfully produces the root word "ukur" correctly.

3.3 Error Analysis

Error analysis was conducted to examine the causes of inaccuracy in the stemming results produced by the Nazief–Adriani and Porter algorithms in Javanese language processing. Although both algorithms have the same objective, namely to convert inflected words into root words, differences in processing mechanisms, particularly in the existence of dictionary checks, affix removal strategies, and minimum word length limits, cause differences in stemming results. The differences in the processing stages performed by each algorithm, as shown in Table 5, have the potential to cause errors in the form of over-stemming and under-stemming, so further analysis is needed to understand the characteristics of errors in both methods.

Table 5. Comparison of Nazief-Adriani and Porter's Methods

No	Process Stages	Nazief–Adriani	Porter
1	Word Normalization	Change letters to lowercase and convert é, è, ê to e.	Change letters to lowercase and convert é, è, ê to e.
2	Initial Length Check	Does not directly limit word length, but the process is controlled in the next stage.	If the word length is ≤ 3 letters, the process is stopped.
3	Initial Dictionary Check	Matches at least the first 3 letters of the word with the dictionary. If found, the process is stopped.	Does not perform dictionary checking at the initial stage.
4	Infix Processing	Removes the infixes in, um, el, er after the first letter in certain patterns (nel-, nin-, num-, ner-, mel-, min-, mum-, mer-), then validates against the dictionary.	Remove the infixes in, um, el, er after the first letter without dictionary validation.
5	Word Length Validation	The processing result must be ≥ 4 letters long, otherwise it is returned to its previous form.	The processing result must be ≥ 3 letters long.
6	Advanced Dictionary Check	The infix result is checked again in the dictionary with a minimum of 3 initial letters.	No dictionary check at this stage.
7	Luluh Processing (Luluh)	Returns phonological changes due to prefixes (n-, m-, ny-, ng-). Specifically, the prefix ma- is softened and then validated in the dictionary.	Identifies and handles softened prefixes directly (ny-, ng-, n-, m-) based on phonological rules.
8	Dictionary Check After Softening	The softened result is checked again against the dictionary with a minimum of 3 initial letters.	No dictionary check.
9	Prefix Removal	Prefixes are removed gradually and selectively. Prefixes are validated against the dictionary; if they fail, they are reprocessed into di-.	Prefixes are removed once based on the prefix list, whether softened or normal.
10	Prefix Length Validation	The result of prefix removal must have a length of ≥ 4 letters before continuing.	The result of prefix removal must have a length of ≥ 3 letters.
11	Dictionary Check After Prefix	The result of prefix removal is matched back to the dictionary.	No dictionary check is performed.

Copyright © 2026 The Author, Page 813
This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

No	Process Stages	Nazief–Adriani	Porter
12	Suffix Removal	Suffixes are removed gradually: -ake (≥4), -ke (≥5), -e (≥4), a maximum of two times.	Remove one suffix from the list of available suffixes if the word length is ≥ 3 letters.
13	Final Dictionary Check	The final result is matched to the dictionary based on at least the first 3 letters.	No final dictionary check.
14	Stemming Result	The last word with a minimum length of 3 letters and valid in the dictionary is set as the base word.	The last word after all rules are applied is set as the base word.

Based on the evaluation results of several sample words, for the word “disusun”, the Nazief–Adriani algorithm produced the form “susu”, which occurred because the form ‘susun’ was not found in the dictionary checking stage, so the process continued until a shorter word was produced. Meanwhile, the Porter Stemmer was able to produce the correct form, “susun”, because the cutting was done purely based on the rules without checking the dictionary. For the word “tumeka”, the Porter Stemmer produced the form ‘tek’ because “-a” was considered a suffix and the process was stopped when the word length reached three letters, while the Nazief–Adriani algorithm was able to maintain a more accurate form due to the dictionary checking stage. Another error is seen in the word “telune”, which the Porter Stemmer processes into “tun”, while the Nazief–Adriani algorithm produces the form “tune” because the infix processing rules at that stage are not strict enough. For the word “mangana”, the Porter Stemmer produced the form ‘nga’ due to the application of the prefix rule “ma-” to words with a minimum length of seven letters and the removal of “-na” which is considered a suffix, while the Nazief–Adriani algorithm was able to produce the base form “pangan” because the prefix “ma-” was first softened and checked against the dictionary, and only if no match was found was the prefix truncated.

3.4 Discussion

This study differs fundamentally from previous studies discussing Javanese stemming. Most previous studies tended to focus on the single application of the Nazief–Adriani algorithm or only compared it with the Porter Stemmer in general, without placing the morphological characteristics of the Javanese language as a major factor in the analysis. In addition, previous studies generally processed prefixes and suffixes, so that the role of infixes, which are quite productive in Javanese, has not been studied in depth. As a result, the source of stemming errors in infixed words and affix combinations is often not specifically identified.

Unlike previous studies, this study includes infix processing as a major part of the Nazief–Adriani and Porter Stemmer algorithms. This approach allows for clearer identification of stemming error patterns that arise due to infixes, particularly in the Porter algorithm, which works without dictionary validation. The results of the study show that the presence of infixes such as -in-, -um-, -el-, and -er, contributes significantly to an increase in over-stemming errors in the Porter Stemmer, while the Nazief–Adriani algorithm tends to be more stable because each cutting result is always validated against the root word dictionary.

In addition, this study not only compares the final accuracy values, but also analyzes stemming errors based on the internal process stages of each algorithm. This analysis was conducted by linking the results of over-stemming and under-stemming errors with the infix, prefix, and suffix cutting stages as shown in the stemming process table. This approach differs from previous studies, which generally stop at reporting evaluation metrics without tracing the causes of errors at each stage. Thus, the contribution of this study lies in revealing the sources of stemming errors in more detail and providing a stronger basis for the development of Javanese stemming algorithms that are more adaptive to its morphological complexity.

3.5 Evaluation

The performance evaluation of the algorithms in this study was conducted using a confusion matrix. This method was used to analyze the success rate of each algorithm in generating the correct base words. The evaluation results were expressed in terms of accuracy, precision, recall, and F1-score values to show the differences in performance between algorithms. The evaluation results can be seen in Table 5.

Table 5. Comparison Results of Nazief-Adriani with Porter Stemmer

Metric	Nazief-Adriani	Porter
Amount of Test Data	603	603
Correct Amount	543	459
Over-stemming	50	116
Under-stemming	10	28
Accuracy (%)	90.05%	76.12%
Precision (%)	98.19%	94.25%
Recall (%)	91.57%	79.83%
F1-Score (%)	94.76%	86.44%

This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

Table 5 above shows that Nazief–Adriani performs better than Porter Stemmer. Nazief–Adriani achieved an accuracy score of 90.05%, while Porter Stemmer only reached 76.12%. This difference in accuracy values indicates that Nazief–Adriani is better able to produce root words that match the dictionary, thereby minimizing stemming errors. Meanwhile, Porter's lower accuracy value indicates that there are still inaccuracies in the cutting of affixes in the Javanese stemming process.

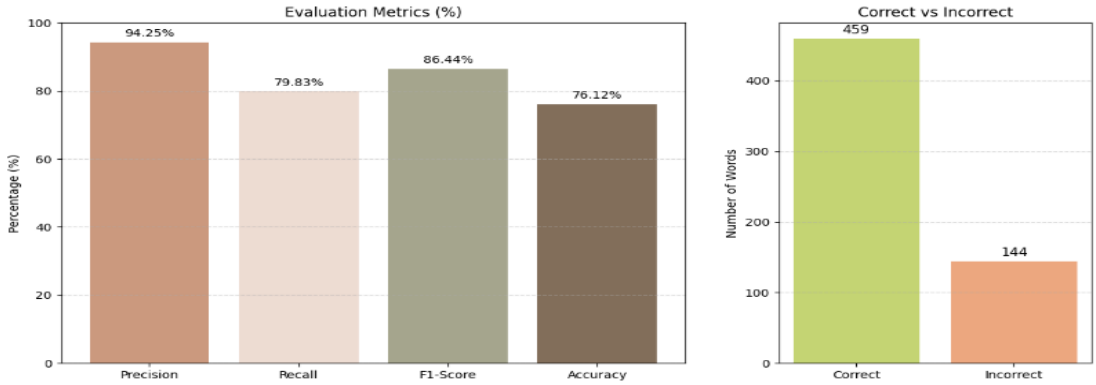

Figure 5. Porter Stemmer Bar Chart

Figure 5 shows the results of testing Porter Stemmer in determining Javanese root words. A precision value of 94.25% indicates that most of the stemming results do not remove root words. However, the recall value of 79.83% indicates that there are still a number of root words that are lost due to excessive truncation. The F1-score value of 86.44% reflects the balance between accuracy and completeness of the results, while the accuracy of 76.12% indicates that about three-quarters of all test data was processed correctly.

In the graph comparing correct and incorrect results, it can be seen that 459 words were successfully stemmed correctly, while 144 words still had errors. This shows that although the Porter Stemmer is quite good at producing the correct root words, this algorithm still has limitations in handling the complex variations of Javanese affixes, so that some words cannot yet be processed optimally.

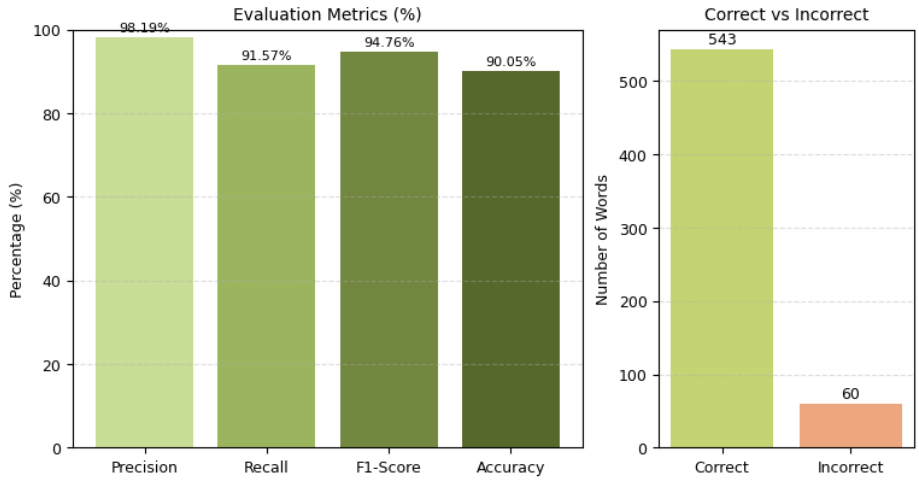

Figure 6. Nazief-Adriani Bar Chart

Figure 6 shows that the Nazief–Adriani algorithm performs quite well with a precision value of 98.19% and a recall of 91.57%, as well as an F1-score of 94.76%. The test results show that most affixed words can be processed correctly by this algorithm. Of the total test data, 543 words were successfully stemmed correctly, while 60 other words still produced errors. When compared to the Porter Stemmer, the number of errors in Nazief–Adriani appears to be fewer, indicating that this algorithm is better able to follow the word formation patterns in Javanese. In general, these results show that Nazief–Adriani provides more consistent performance in the stemming process.

4. CONCLUSION

Based on the research results, it can be concluded that the Nazief–Adriani algorithm demonstrates better performance than the Porter Stemmer in the Javanese language stemming process. This is evidenced by an accuracy rate of 90.05% for the Nazief–Adriani algorithm, whereas the Porter Stemmer only achieved 76.12%.

Copyright © 2026 The Author, Page 815
This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

Furthermore, the precision, recall, and F1-score values produced by the Nazief–Adriani algorithm are also higher, indicating that this algorithm is more consistent in generating the correct base forms. This difference in performance is influenced by the distinct approaches employed by each algorithm. The Nazief–Adriani algorithm uses a dictionary-based approach combined with morphological rules, so that every result of affix removal is verified against a base word dictionary. Meanwhile, Porter only performs affix removal based on existing rules, which often leads to errors in determining the base word. Thus, Nazief–Adriani has proven effective in handling the morphological complexity of the Javanese language, including phonological changes and the presence of affix and infix combinations. Moving forward, further development could focus on expanding and refining the Javanese root word dictionary, as well as creating stricter and more structured infix processing rules—such as infixes being removed only when they appear with a vowel—so that the stemming process can produce more accurate and stable root words. In conclusion, the findings of this study highlight the importance of selecting a stemming algorithm that is appropriate and tailored to the linguistic characteristics of the Javanese language. By taking into account both morphological complexity and dictionary validation, the Nazief–Adriani algorithm demonstrates higher reliability and effectiveness in processing Javanese text. Therefore, this algorithm is recommended for research involving Javanese natural language processing tasks.

REFERENCES

[1] K. Gita Segara and M. Irwan Padli Nasution, “Perkembangan Teknologi Informasi di Indonesia: Tantangan dan Peluang,” Jurnal Sains Student Research, vol. 3, no. 1, pp. 21–33, 2025, doi: 10.61722/jssr.v3i1.3128.
[2] O. : Prof and R. C. Tarumingkeng, “Natural Language Processing (NLP),” 2024.
[3] S. Pais, J. Cordeiro, and M. L. Jamil, “NLP-based platform as a service: a brief review,” J. Big Data, vol. 9, no. 1, Dec. 2022, doi: 10.1186/s40537-022-00603-5.
[4] F. S. P. Azzahra and C. Prianto, “Tinjauan Literatur Sistematis: Analisis Implementasi Kecerdasan Buatan untuk Verifikasi Dokumen,” Jurnal Ilmiah Sistem Informasi, vol. 4, no. 3, pp. 417–430, Aug. 2025, doi: 10.51903/kjjwk708.
[5] D. Arbian Sulistyo et al., “LSTM-Based Machine Translation for Madurese-Indonesian,” Journal of Applied Data Sciences, vol. 4, no. 3, pp. 190–199, 2023.
[6] D. Sulistyo, F. Ahda, and V. A. Fitria, “Epistomologi dalam Natural Language Processing,” Jurnal Inovasi Teknologi dan Edukasi Teknik, vol. 1, no. 9, pp. 652–664, Sep. 2021, doi: 10.17977/um068v1i92021p652-664.
[7] F. A. Ahda, A. P. Wibawa, D. D. Prasetya, D. A. Sulistyo, and A. Nafalski, “Minangkabau Language Stemming: A New Approach with Modified Enhanced Confix Stripping,” Jurnal RESTI, vol. 9, no. 3, pp. 677–687, Jun. 2025, doi: 10.29207/resti.v9i3.6511.
[8] Ja’a Nazielatu Rois Nabilla and Fitri Anekawati, “Afiksasi pada Bahasa Jawa, Bahasa Inggris, dan Bahasa Indonesia,” Sintaksis : Publikasi Para ahli Bahasa dan Sastra Inggris, vol. 3, no. 3, pp. 245–252, Apr. 2025, doi: 10.61132/sintaksis.v3i3.2219.
[9] E. P. Asri, J. Findy Addila, and J. Aprillia, “Sabdasastra: Jurnal Pendidikan Bahasa Jawa PROSES AFIKSASI CRIKAK ‘DUDU RAMA LAN SHINTA’ KARYA SURYADI WS. SEBAGAI PEMAHAMAN MAKNA KATA BERBAHASA JAWA,” Jurnal Pendidikan Bahasa Jawa, vol. 9, no. 2, pp. 175–184, 2025, doi: 10.20961/sabpbj.v9i2.
[10] W. D. Suryono, E. Utami, and D. Ariatmanto, “Analisa Perbandingan Stemming Dokumen Teks Berbahasa Jawa dengan Algoritma Levenshtein Distance Dan Jaro-Winkler,” JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika), vol. 10, no. 1, pp. 774–780, Jan. 2025, doi: 10.29100/jipi.v10i1.6092.
[11] A. P. Wibawa and M. N. Hakim, “STEMMING BAHASA JAWA MENGGUNAKAN DAMERAU LEVENSHTEIN DISTANCE (DLD),” JURNAL TEKNIK INFORMATIKA, vol. 14, no. 1, pp. 22–27, Sep. 2021, doi: 10.15408/jti.v14i1.15010.
[12] St Tuhpatussania and S. Erniwati, “PERBANDINGAN ALGORITMA NAZIEF-ADRIANI DAN PORTER UNTUK PERINGKASAN TEKS OTOMATIS DENGAN LATENT SEMANTIC ANALYSIS PADA MODUL PEMBELAJARAN BERBAHASA INDONESIA,” Jurnal Kecerdasan Buatan dan Teknologi Informasi, vol. 4, no. 1, pp. 53–59, Jan. 2025, doi: 10.69916/jkbti.v4i1.202.
[13] A. P. Wibawa, F. A. Dwiyanto, I. A. E. Zaeni, R. K. Nurrohman, and A. Afandi, “Stemming javanese affix words using nazief and adriani modifications,” Jurnal Informatika, vol. 14, no. 1, p. 36, Jan. 2020, doi: 10.26555/jifo.v14i1.a17106.
[14] D. Soyusiawaty, A. H. S. Jones, and N. L. Lestariw, “The Stemming Application on Affixed Javanese Words by using Nazief and Adriani Algorithm,” in IOP Conference Series: Materials Science and Engineering, Institute of Physics Publishing, Mar. 2020. doi: 10.1088/1757-899X/771/1/012026.
[15] S. Tuhpatussania, E. Utami, and A. D. Hartanto, “COMPARISON OF PORTERS STEMMING ALGORITHM AND NAZIEF & ADRIANI’S STEMMING ALGORITHM IN DETERMINING INDONESIAN LANGUAGE LEARNING MODULES,” Jurnal Pilar Nusa Mandiri, vol. 18, no. 2, pp. 203–210, Sep. 2022, doi: 10.33480/pilar.v18i2.3940.
[16] Ayunda Putri Nazarrina, Annisa Dallilatul Hasanah, Muhamad Ipnu Hasan, and Hasan Suaedi, “Analisis Morfofonemis Prefiks dalam Percakapan Dialog pada Buku Sekkar Assre’ 4,” Semantik : Jurnal Riset Ilmu Pendidikan, Bahasa dan Budaya, vol. 3, no. 1, pp. 255–267, Jan. 2025, doi: 10.61132/semantik.v3i1.1430.
[17] Moh Ashari, D. A. Sulistyo, and F. A. Ahda, “STEMMING IN MADURESE LANGUAGE USING NAZIEF AND ADRIANI ALGORITHM,” Jurnal Teknik Informatika (Jutif), vol. 5, no. 4, pp. 695–702, Aug. 2024, doi: 10.52436/1.jutif.2024.5.4.2012.
[18] S. I. Melia, J. Sholihah, D. Nisak, I. S. Juniaristha, and A. T. Ni’mah, “The Ngoko Javanese Stemmer uses the Enhanced Confix Stripping Stemmer Method,” Rekayasa, vol. 16, no. 1, pp. 107–112, Apr. 2023, doi: 10.21107/rekayasa.v16i1.19308.
[19] R. Aulia, I. Baehaqie, and R. Rustono, “Proses Morfofonemik pada Bahasa Jawa Kabupaten Bengkalis,” GERAM, vol. 11, no. 2, pp. 134–141, Dec. 2023, doi: 10.25299/geram.2023.vol11(2).15293.

This Journal is licensed under a Creative Commons Attribution 4.0 International License

Journal of Information System Research (JOSH)
Volume 7, No. 4, Juli 2026, pp 808−817
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i4.9019

[20] A. Rio Hintono, A. Subiyanto, J. Soedarto, and K. Undip Tembalang, "Proses Morfofonemik pada Pembentukan Verba dengan Sufiks '-i' dalam Bahasa Jawa," ANUVA, vol. 7, no. 1, pp. 14–28, 2023.
[21] Z. Abidin, A. Junaidi, and Wamiliana, "Text Stemming and Lemmatization of Regional Languages in Indonesia: A Systematic Literature Review," Journal of Information Systems Engineering and Business Intelligence, vol. 10, no. 2, pp. 217–231, Jun. 2024, doi: 10.20473/jisebi.10.2.217-231.
[22] Y. Trisniwati and S. Dwi, BUKU PINTER PEPAK BAHASA JAWA LENGKAP EDISI REVISI.
[23] J. Pardede and D. Darmawan, "PERBANDINGAN ALGORITMA STEMMING PORTER, SASTRAWI, IDRIS, DAN ARIFIN & SETIONO PADA DOKUMEN TEKS BAHASA INDONESIA," vol. 12, no. 1, pp. 69–76, 2025, doi: 10.25126/jtiik.2025128860.
[24] D. Arbian Sulistyo et al., "Multilingual Parallel Corpus for Indonesian Low-Resource Languages," INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION, 2025, doi: http://dx.doi.org/10.62527/joiv.9.5.3412.

This Journal is licensed under a Creative Commons Attribution 4.0 International License