N-Gram Stemming for Dialect of *Api* Lampung

1st Zaenal Abidin
Doctoral of Mathematics and Natural Sciences
Universitas Lampung
Bandar Lampung, Indonesia
2237061006@students.unila.ac.id

2nd Akmal Junaidi
Department of Computer Science
Universitas Lampung
Bandar Lampung, Indonesia
akmal.junaidi@fmipa.unila.ac.id

3rd Wamiliana
Department of Mathematics
Universitas Lampung
Bandar Lampung, Indonesia
wamiliana.1963@fmipa.unila.ac.id

4th Favorisen R.Lumbanraja
Department of Computer Science
Universitas Lampung
Bandar Lampung, Indonesia
favorisen.lumbanraja@fmipa.unila.ac.id

5th Dian Kurniasari
Department of Mathematics
Universitas Lampung
Bandar Lampung, Indonesia
dian.kurniasari@fmipa.unila.ac.id

6th Rohmat Indra Borman
Faculty of Engineering and Computer Science
Universitas Teknokrat Indonesia
Bandar Lampung, Indonesia
rohmat_indra@teknokrat.ac.id

2025 4th International Conference on Electronics Representation and Algorithm (ICERA) | 979-8-3315-9582-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICERA66156.2025.11087285

***Abstract*— Stemming is the technique of extracting base words from affixed words to improve recall by minimizing the variance of affixed words down to their base word forms. We used the n-gram stemming method, in which the affixed and base words are converted to n-gram form, and the degree of similarity between the n-gram of the affixed word and the n-gram of the base word is measured using the dice coefficient method; if the degree of similarity meets the specified threshold value, the base word is displayed in comparison to the affixed word. In this study, we created a stemmer approach that combines all affix variants in the Lampung language dialect of *Api* using the n-gram stemming method. At first, we looked for the best threshold value for the N-Gram Stemming approach, which we found to be 0.5 for bigram and trigram. Based on 500 test words, the Bigram accuracy score is 0.94 for a threshold of 0.5 and 0.91 for a threshold of 0.55, but the Trigram accuracy value is 0.88 for a threshold of 0.5 and 0.79 for a threshold of 0.55. According to the results from 200 independent test words, the Bigram accuracy value is 0.93 for threshold 0.5 and 0.89 for threshold 0.55, while the Trigram accuracy value is 0.87 for threshold 0.5 and 0.80 for threshold 0.55. In this study, we were focused on the morphology of each word and the validation stage with a dictionary.**

Keywords*— *N-Gram Stemming, Bigram, Trigram, Threshold, Dialect of Api

I. Introduction

Lampung *Api* is one of the main dialects of the Lampung language, a regional language is spoken by people in Lampung Province, Indonesia, particularly in coastal areas such as Bandar Lampung, Pesawaran, Pringsewu, and parts of Tanggamus, in contrast to the more common *Nyo* dialect of Lampung inland; "*Api*" means "what" in this dialect, reflecting its characteristic vocabulary and intonation that distinguishes it from the *Nyo* dialect which uses "*nyo*" for similar questions [1]. At the same time, the language also has a traditional script called *Had Lampung*, although it is now more often written with Latin letters in everyday life. Lampung is one of the regional languages in the archipelago, and its speakers still maintain it. It functions as an intra-tribal lingua franca. In addition to this function, Lampung is still used as the language of instruction in traditional ceremonies, such as wedding ceremonies, naming, and giving titles and circumcisions [2][3].

Stemming is a process that correlates the morphological variants of a word to its base form. Stemming is widely employed as a preprocessing technique in natural language processing, information retrieval, and language modeling. Despite significant breakthroughs, a systematic organization of prior work and efforts remains deficient. Singh and Gupta comprehensively studied text-stemming theory, techniques, and applications [4][5]. The work initially reviews the pertinent literature on text stemming, categorizing it based on several essential features; thereafter, Singh and Gupta provide an in-depth study of several prominent stemming algorithms using standard data sets [4][5]. The present state-of-the-art and specific unresolved issues concerning unsupervised stemming are ultimately delineated. Singh and Gupta stated that the primary objective of this study is to furnish a comprehensive and valuable insight into the critical elements of text stemming [4][5]. The existing challenges and examination of current stemming procedures will assist researchers in identifying new avenues for future investigation.

In the literature, various performance evaluation criteria exist for stemmers, each assessing performance from a particular perspective. In [6], the authors evaluated and analyzed text-stemming evaluation methodologies to establish criteria for improved measurement of stemmer performance. The discussion focuses on several facets of stemmer performance evaluation, including primary characteristics, advantages, and limitations, utilizing a resource-scarce language, specifically Urdu [6]. Their investigations indicated that the existing evaluation measures could solely assess an average conflation of words, irrespective of the accuracy of the stem [6]. Furthermore, many evaluation measures preferentially benefit specific language kinds. No current evaluation metrics can accurately assess stemmer performance across all language types.

Stemming or lemmatization research on regional languages in Indonesia has begun, and stemming research in Indonesian has been pioneered [7][8][9]. Stemming or lemmatization research in local languages is based on morphological analysis and the use of local language dictionaries. Stemming in Javanese [10], [11], [12], [13], Balinese [14], [15], [16], [17], [18], Madurese [19], [20], [21], Sundanese [22], [23], [24], [25], Rejang [26], [27],

Minangkabau [28], [29], Riau Malay [30], Lampung [31], [32], and Tetun [33]. Several SLR publications related to stemming or lemmatization are also available, making it easier for readers to know the SOTA of stemming or lemmatization research [34], [35]. This research, especially for the Lampung language, continues previous research [31], [32]. Adamson and Boreham developed the n-gram stemming method, demonstrating that words with more remarkable structural similarity often share similar meanings [15].

In this approach, affixed words and base words are transformed into n-gram representations, and the similarity between the n-grams of the affixed word and the base word is assessed using the Dice coefficient method. If the similarity meets the predetermined threshold, the base word about the affixed word is presented [15]. The primary issue in this research is the direct machine translation application's inability to translate affixed words accurately in the Lampung dialect of *Api*. Various stemming methods are employed in different regional languages to handle affixed words, with the N-Gram stemming method being used in the Lampung language. The main contribution is to convert the affixed words into basic words, allowing for word matching between the results and the vocabulary available in the digital dictionary.

II. N-GRAM STEMMING METHOD

A. N-Gram Stemming

If the available rules cannot recognize the affixed word, then the string similarity process uses the n-gram stemming method [15]. The characteristics of affixed words that cannot be recognized by the rules, one of which is due to errors in writing affixed words, can be overcome by N-Gram Stemming [15]. The process flow of the n-gram stemming method is shown in Figure 1.

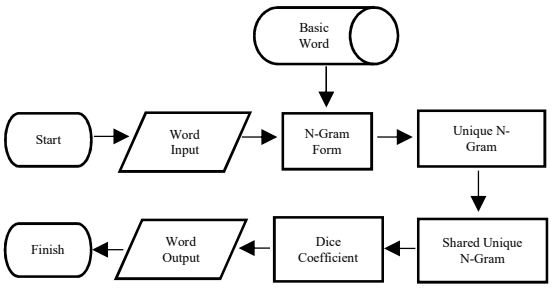

Fig. 1. Flow Process of N-Gram Stemming [15]

Initially, the affixed word and the root word in the database are transformed into n-gram format, followed by a comparison that involves tallying the number of unique n-grams or the count of n-gram characters generated and the shared unique n-grams, or identical n-gram character counts between the affixed word and the root word. The similarity between n-grams of attached words and n-grams of base words is quantified using the Dice coefficient approach as per equation (1) [15].

$$dc = (2 . c) / (a + b) \quad (1)$$

In equation (1), *dc* represents the dice coefficient. *c* represents the shared unique n-gram between two words, *a* indicates the unique n-gram of the input word, and *b* indicates the unique n-gram of the root word [15]. The process of assessing the similarity between two words, *a* from the input word and *b* from the root word, uses n-gram stemming, using the number of characters n, which is two as bi-grams or three as tri-grams [15].

B. Establishing Threshold Value

When the outcome of the n-gram stemming computation between the input word and the base word satisfies the threshold value, the root word is presented. The threshold values employed are 0.70, 0.65 and 0.60 [15]. This research aimed to identify the optimal threshold value between 0.5 and 0.8. The optimal outcome was achieved at 0.5.

III. RESEARCH METHODOLOGY

The research stages denote the sequential steps undertaken in the research process to achieve the intended outcomes. The procedures undertaken are (1) data acquisition, (2) algorithm formulation, (3) execution and assessment, and (4) results evaluation. The research stages are depicted in Fig. 2.

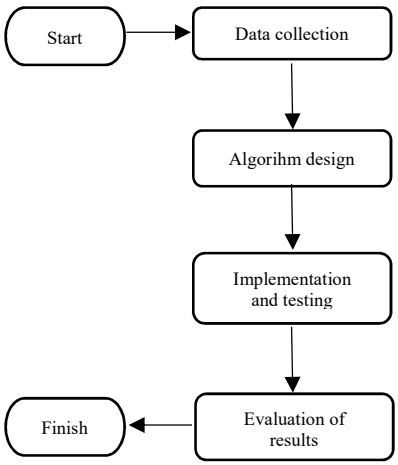

Fig. 2. The research stages

The stages of research about the N-Gram Stemming stemmer in the dialect of *Api* are as follows:

- *Data collection* : This study gathers data using essential word dictionary data, test word data, and independent test word data. Manually inputting 7669 terms into the Lampung-Indonesian Dictionary produced the fundamental lexical information. The dataset utilized for evaluating the N-Gram stemming comprises 500 test words and 200 independent test words, as referenced in the book *Tata Bahasa Bahasa Lampung Dialek Pesisir* / *Api* and *Bahasa Lampung untuk Sekolah Dasar*.
- Algorithm design: The formulation of the algorithm relates to the N-Gram stemming algorithm based on how N-Gram Stemming works. The design phase employs detailed pseudocode. The pseudocode for the N-Gram Stemming dialect of Api is presented in Fig. 3.
- *Implementation and Testing*: Implementation refers to the application of the algorithm derived from the preceding stage. The pseudocode implementation was executed in Python on Google Colab.

- *Evaluation of results*: Stemming evaluation employing gold standard assessment is a technique for gauging the quality and precision of stemming algorithms by contrasting the algorithm's stemmed outputs with a compilation of accurate base words (gold standard).

```
CLASS NGramStemmer:
    FUNCTION initialize(dictionary_file: string):
        """Initialize stemmer with dictionary from CSV"""
        TRY:
            READ CSV file dictionary_file
            CREATE dictionary as SET of lowercase words from 'kata' column
        CATCH error:
            DISPLAY "Error loading dictionary: " + error
            SET dictionary to empty set
    FUNCTION get_ngrams(word: string, n: integer) -> list:
        """Generate n-grams from word"""
        CONVERT word to lowercase
        IF length of word < n THEN
            RETURN empty list
        RETURN [word substring from i to i+n for i in range(word length - n + 1)]
    FUNCTION calculate_similarity(word1: string, word2: string, n: integer) -> float:
        """Calculate similarity using n-grams"""
        SET ngrams1 = set of get_ngrams(word1, n)
        SET ngrams2 = set of get_ngrams(word2, n)
        IF ngrams1 is empty OR ngrams2 is empty THEN
            RETURN 0.0
        SET intersection = size of (ngrams1 INTERSECT ngrams2)
        SET union = size of ngrams1 + size of ngrams2
        IF union > 0 THEN
            RETURN (2 * intersection) / union
        ELSE
            RETURN 0.0
    FUNCTION find_root_word(word: string, n: integer, threshold: float) -> tuple:
        """Find root word using specified n-gram size and threshold"""
        SET best_score = 0.0
        SET best_match = word

        FOR each dict_word IN dictionary:
            SET score = calculate_similarity(word, dict_word, n)
            IF score > best_score AND score >= threshold THEN
                SET best_score = score
                SET best_match = dict_word
        RETURN tuple(best_match, best_score)
FUNCTION process_words(stemmer: NGramStemmer, test_file: string):
    """Process test words with specific n-gram sizes and thresholds"""
    TRY:
        // Read test data
        READ CSV file test_file
        GET first_column_name from test data

        // Specific configurations
        SET configurations = [
            {'n': 2, 'threshold': 0.50},  // Bigram with 0.50 threshold
            {'n': 2, 'threshold': 0.55},  // Bigram with 0.55 threshold
            {'n': 3, 'threshold': 0.50},  // Trigram with 0.50 threshold
            {'n': 3, 'threshold': 0.55}   // Trigram with 0.55 threshold
        ]
        CREATE empty all_results list
        FOR each config IN configurations:
            SET n = config['n']
            SET threshold = config['threshold']
            DISPLAY "Processing {n}-grams with threshold {threshold}"
            FOR each word IN test_data[first_column_name]:
                CONVERT word to lowercase string
                SET predicted, score = stemmer.find_root_word(word, n, threshold)
                ADD to all_results:
                    test_word = word
                    root_word = predicted
                    similarity_score = score
                    n_gram_size = n
                    threshold = threshold
        // Save results to Excel
        CREATE Excel file 'ngram_stemming_specific_results.xlsx'
       // Save all results
        WRITE all_results to sheet 'All Results'
       // Create separate sheets for each configuration
        FOR each config IN configurations:
            SET sheet_name = '{n}-gram_t{threshold}'
            FILTER results for this configuration
            WRITE results to sheet sheet_name
        // Create summary sheet
        CREATE empty summary_data list
        FOR each config IN configurations:
            FILTER subset data for this configuration
            ADD to summary_data:
                n_gram_size = n
                threshold = threshold
                total_words = word count
                unique_roots = unique root word count
                avg_similarity = average score
                min_similarity = minimum score
                max_similarity = maximum score
        // Save summary
        WRITE summary_data to sheet 'Summary'
        DISPLAY "Results saved to ngram_stemming_specific_results.xlsx"
        DISPLAY "Summary of results:"
        DISPLAY summary
    CATCH error:
        DISPLAY "Error processing words: " + error

FUNCTION main():
    // File paths
    SET dictionary_file = 'KKDKBLLL.csv'
    SET test_file = '500katauji1.csv'
    // Initialize stemmer
    CREATE stemmer = NGramStemmer(dictionary_file)
    // Process words with specific configurations
    CALL process_words(stemmer, test_file)
IF this is main program THEN
    CALL main()
```

Fig. 3. The Pseudocode of N-Gram Stemming

A. Details of the Technical Stages of N-Gram Stemming

N-gram stemming is a natural language processing technique comprising five systematic steps to reduce words to their root forms.

1. The process begins with data preparation, including collecting words to be stemmed, establishing a basic word dictionary as a reference, and setting a threshold value of 0.5 as a similarity benchmark.
2. The second step employs the bi-gram method (n=2) by breaking words into pairs of two consecutive characters, doing the same for dictionary words, and then calculating similarity using Dice's Coefficient formula, where words are considered similar if their similarity value exceeds 0.5.
3. Concurrently, the third step applies the tri-gram process (n=3) that segments words into sets of three consecutive characters compares them with dictionary words using the same method, and verifies whether the similarity value exceeds the 0.5 thresholds.
4. Next, the decision-making step selects the base word with the highest similarity value (exceeding 0.5) from either the bi-gram or tri-gram method. The system chooses the shortest if several words share the same similarity value.
5. Finally, the stemming result is determined by transforming words with similarity values greater than 0.5 into their root forms according to the dictionary reference, while words with similarity values less than or equal to 0.5 remain in their original form, making the entire process an effective method for normalizing morphological variations of words in natural language processing applications.

This process helps identify words with structural similarities and can be used to more accurately find the base word of a compound word.The stemming accuracy results are computed using equation (2) [J].

$$\textit{Gold standar assessment} = (t / n) \text{ x } 100 \qquad (2)$$

In equation (2), the gold standard assessment represents the stemming accuracy. t represents the quantity of words that have been accurately stemmed, and n represents the quantity of fastened words.

IV. Result and Discussion

N-Gram stemming is developed without going through an exploration based on morphological rules but using a language dictionary. Experiments were conducted focusing on the environments (1) bigram with threshold 0.5 and 0.55 and (2) trigram with threshold 0.5 and 0.55. The Python code was then tested on the Google Colab environment for implementation. The input consisted of 500 test words and 200 independent test words. All proofs of this experiment are archived and can be accessed at the link bit.ly/42obnqM.

A. Experimental Findings on 500 Test Words

In the Bi-Gram Stemming method with a threshold of 0.5, the test results show that out of a total of 500 words tested, 30 words fail to stem (Failed Stemming), while 470 words are successfully stemmed (Successful Stemming). Thus, the accuracy of this method reaches 0.94 or 94%. Table I shows that the Bi-Gram Stemming method, with a threshold of 0.5,

has a fairly high success rate in identifying the basic form of words in *Api* dialects. The stemming performance decreases slightly when the threshold is increased to 0.55 in the Bi-Gram Stemming method. Out of 500 words tested, 45 words failed to stem, while 455 were successfully stemmed. The accuracy of this method is 0.91 or 91%, which is lower than the threshold of 0.5. This shows that increasing the threshold in Bi-Gram Stemming tends to increase the number of stem failures, which may be due to stricter criteria for matching words.

TABLE I. BI-GRAM STEMMING RESULT IN 500 DATA TEST

Test Word	Bi-Gram Stemming	
	Threshold 0.5	Threshold 0.55
Failed Stemming Words	30	45
Successful Stemming Words	470	455
Total Words	500	500
Accuracy	0.94	0.91

TABLE II. TRI-GRAM STEMMING RESULT IN 500 DATA TEST

Test Word	Tri-Gram Stemming	
	Threshold 0.5	Threshold 0.55
Failed Stemming Words	59	107
Successful Stemming Words	441	393
Total Words	500	500
Accuracy	0.882	0.786

The Tri-Gram Stemming method with a threshold of 0.5 shows lower performance than Bi-Gram Stemming. Out of a total of 500 words, 59 words failed to be stemmed, and 441 words were successfully stemmed. The accuracy of this method is 0.882 or 88.2%. Although the accuracy is still quite good, this method shows a higher failure rate than Bi-Gram Stemming at the same threshold, which may be due to the Tri-Gram approach, which is more complex in grouping words. When the threshold of the Tri-Gram Stemming method is increased to 0.55, the stemming performance decreases. Of the 500 words tested, 107 failed to be stemmed, while 393 were successfully stemmed. The accuracy of this method is 0.786 or 78.6%, the lowest accuracy among all tested methods. Increasing the threshold in Tri-Gram Stemming seems to significantly impact the failure rate, indicating that this method is more sensitive to threshold changes than Bi-Gram Stemming.

Overall, Bi-Gram Stemming with a threshold of 0.5 showed the best performance with 94% accuracy, followed by Bi-Gram Stemming with a threshold of 0.55 (91%), Tri-Gram Stemming with a threshold of 0.5 (88.2%), and Tri-Gram Stemming with threshold 0.55 (78.6%). Bi-Gram Stemming is consistently superior to Tri-Gram Stemming at both threshold values, which suggests that the Bi-Gram approach may be more suitable for coastal dialects. In addition, increasing the threshold from 0.5 to 0.55 in both methods decreased accuracy, with a more significant decrease in Tri-Gram Stemming (88.2% to 78.6%) than in Bi-Gram Stemming (94% to 91%).

These results show that choosing the right stemming method and threshold value is crucial in natural language processing, especially for coastal dialects. Bi-gram stemming with a threshold of 0.5 can be the best choice for applications requiring high stemming accuracy. However, adjustments to the threshold and method must be considered further for some instances that may require a stricter approach.

To compare the accuracy based on the information in Tables 1 and 2, a bar chart is presented below.

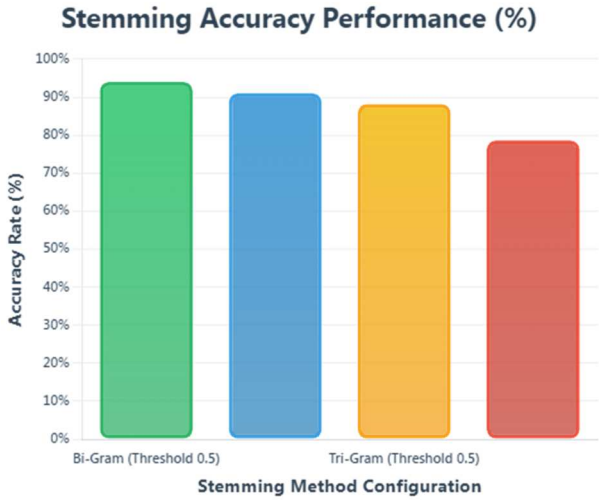

Fig. 4. Stemming Accuracy Performance from Table I and Table II

B. *Experimental Findings on 200 Independent Test Words*

Independent stemming tests were conducted on the *Api* dialect using the N-Gram Stemming method, which is divided into two approaches: Bi-Gram Stemming and Tri-Gram Stemming. Each approach was tested with two threshold values, 0.5 and 0.55, to evaluate its effectiveness in identifying the base form of words. A total of 202 words were tested, and the results were categorized into two main parameters: the number of stemming failures (Failed Stemming) and the number of successes (Successful Stemming). The accuracy of each method was also calculated to give an idea of the overall performance.

TABLE III. BI-GRAM STEMMING RESULT IN 200 INDEPENDENT DATA TEST

Test Word	Bi-Gram Stemming	
	Threshold 0.5	Threshold 0.55
Failed Stemming Words	14	21
Successful Stemming Words	188	181
Total Words	202	202
Accuracy	0.9307	0.8960

In the Bi-Gram Stemming method with a threshold of 0.5, out of 202 words tested, 14 failed to stem (Failed Stemming), while 188 were successfully stemmed (Successful Stemming). The accuracy of this method reaches 0.930693069 or about 93.07%. Table III shows that Bi-Gram Stemming, with a threshold of 0.5, performs well in identifying the basic form of words in coastal fire dialect, with a relatively low failure rate. The stemming performance decreases slightly when the threshold is increased to 0.55 in the Bi-Gram Stemming method. Of the 202 tested, 21 words failed to be stemmed,

while 181 were successfully stemmed. The accuracy of this method is 0.896039604 or about 89.60%, which is lower than the threshold of 0.5. Increasing the threshold seems to cause more failed words, which may be due to the stricter matching criteria.

In the Tri-Gram Stemming method with a threshold of 0.5, stemming performance shows lower results than Bi-Gram Stemming. From a total of 202 words, 27 words fail to stem, and 175 words successfully stem. The accuracy of this method is 0.866336634 or about 86.63%. Although the accuracy is still quite good, this method has a higher failure rate than Bi-Gram Stemming at the same threshold, which may be due to the Tri-Gram approach, which is more complex in grouping words. When the threshold of the Tri-Gram Stemming method is increased to 0.55, the stemming performance decreases. Of the 202 words tested, 40 failed to be stemmed, while 162 were successfully stemmed. The accuracy of this method is 0.801980198 or about 80.20%, which is the lowest accuracy among all tested methods. Increasing the threshold in Tri-Gram Stemming significantly impacts the failure rate, indicating that this method is more sensitive to threshold changes than Bi-Gram Stemming.

TABLE IV. TRI-GRAM STEMMING RESULT IN 200 INDEPENDENT DATA TEST

Test Word	Tri-Gram Stemming	
	Threshold 0.5	Threshold 0.55
Failed Stemming Words	27	40
Successful Stemming Words	175	162
Total Words	202	202
Accuracy	0.87	0.80

Overall, Bi-Gram Stemming with a threshold of 0.5 shows the best performance with 93.07% accuracy, followed by Bi-Gram Stemming with a threshold of 0.55 (89.60%), Tri-Gram Stemming with a threshold of 0.5 (86.63%), and Tri-Gram Stemming with threshold 0.55 (80.20%). This pattern is consistent with previous tests on coastal dialects, where Bi-Gram Stemming is consistently superior to Tri-Gram Stemming at both threshold values. Increasing the threshold from 0.5 to 0.55 in both methods also tends to decrease accuracy, with a more significant decrease in Tri-Gram Stemming (from 86.63% to 80.20%) compared to Bi-Gram Stemming (from 93.07% to 89.60%).

Compared to the previous testing data on Api dialects (with a total of 500 words), the stemming performance on coastal fire dialects shows a similar pattern. In the previous data, Bi-Gram Stemming with a threshold of 0.5 also had the highest accuracy (94%), while Tri-Gram Stemming with a threshold of 0.55 had the lowest accuracy (78.6%). However, the overall accuracy of this independent test data is slightly lower, possibly due to the different characteristics of coastal fire dialects or the smaller data size (202 words compared to 500 words). Nevertheless, Bi-Gram Stemming, with a threshold of 0.5, remains the most effective method for both types of Api dialects.

These test results confirm that Bi-Gram Stemming with a threshold of 0.5 is the most optimal method for stemming in coastal dialects, both in standard dialects and coastal fire dialects. This approach provides a good balance between accuracy and low failure rate. Meanwhile, Tri-Gram Stemming, especially with a threshold of 0.55, tends to be less effective and may be more suitable for dialects or languages with different structures. For natural language processing applications in coastal dialects, it is recommended to use Bi-Gram Stemming with a threshold of 0.5 as the primary method. However, if further analysis of other dialects or different contexts is required, adjustments to the method and threshold can be considered.

To compare the accuracy based on the information in Tables 3 and 4, a bar chart is presented below.

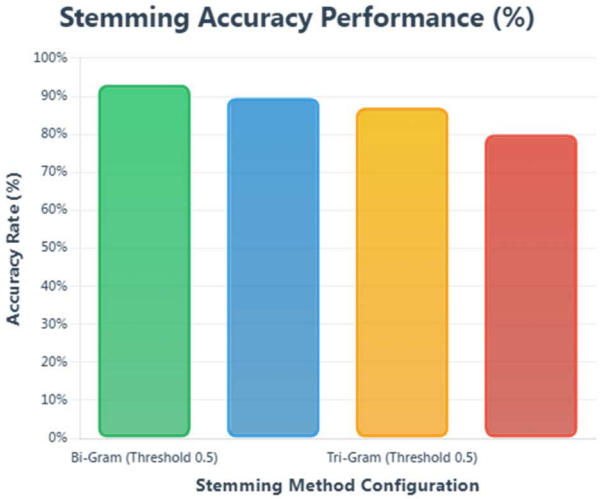

Fig. 5. Stemming Accuracy Performance from Table III and IV

V. CONCLUSION

Bi-Gram Stemming with a threshold of 0.5 emerges as the most effective method for stemming based on the provided data, offering the highest accuracy in both datasets. The use of a higher threshold (0.55) consistently reduces performance, suggesting that it may be too stringent for practical use in this context. Tri-Gram Stemming, while less effective, might benefit from further investigation to understand why it underperforms—possibly due to the complexity of its character triplet-based rules or the nature of the test words. For optimal stemming performance, Bi-Gram Stemming with a threshold of 0.5 is recommended, though additional analysis could help refine these methods further.

ACKNOWLEDGMENT

We would like to acknowledge and express our gratitude to Universitas Teknokrat Indonesia for their contribution on a Hiputs research project which was done under the contract number 006/UTI/LPPMI/E.1.1/VIII/2023.

REFERENCES

[1] M. R. Sudrajat, N. N. WEtty, H. Hadikusumah, and Z. B. Chandau, “Morlologi dan Slntaksls Bahasa Lampung,” 1985.

[2] F. Ariyani, N. E. Rusminto, Sumarti, A. R. Idris, and L. Misliani, “Examining the Forms and Variations of the Lampung Script in Ancient Manuscripts,” *WSEAS Trans. Environ. Dev.*, vol. 18, pp. 204–217, 2022, doi: 10.37394/232015.2022.18.22.

[3] F. Ariyani, G. E. Putrawan, A. R. Riyanda, A. R. Idris, L. Misliani, and R. Perdana, “Technology and minority language: an Android-based dictionary development for the Lampung language maintenance in Indonesia,” *Tapuya Lat. Am. Sci. Technol. Soc.*, vol. 5, no. 1, 2022, doi: 10.1080/25729861.2021.2015088.

[4] J. Singh and V. Gupta, “A systematic review of text stemming

techniques," *Artif. Intell. Rev.*, vol. 48, no. 2, pp. 157–217, Aug. 2017, doi: 10.1007/s10462-016-9498-2.

[5] J. Singh and V. Gupta, "Text stemming: Approaches, applications, and challenges," *ACM Comput. Surv.*, vol. 49, no. 3, Sep. 2016, doi: 10.1145/2975608.

[6] A. Jabbar, S. Iqbal, M. I. Tamimy, S. Hussain, and A. Akhunzada, "Empirical evaluation and study of text stemming algorithms," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5559–5588, Dec. 2020, doi: 10.1007/s10462-020-09828-3.

[7] J. Asian, H. E. Williams, and S. M. M. Tahaghoghi, "Stemming Indonesian," in *Conferences in Research and Practice in Information Technology Series*, 2005, pp. 307–314. doi: 10.1145/1316457.1316459.

[8] A. Z. Arifin, H. T. Ciptaningtyas, P. Adhi, and K. Mahendra, "Enhanced confix stripping stemmer and ants algorithm for classifying news document in indonesian language." In *The International Conference on Information & Communication Technology and Systems*, vol. 5, pp. 149-158. 2009.

[9] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimed. Tools Appl.*, vol. 82, no. 3, pp. 3713–3744, Jan. 2023, doi: 10.1007/s11042-022-13428-4.

[10] K. Swain and A. K. Nayak, "A Review on Rule-Based and Hybrid Stemming Techniques," in *Proceedings - 2nd International Conference on Data Science and Business Analytics, ICDSBA 2018*, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 25–29. doi: 10.1109/ICDSBA.2018.00012.

[11] F. Amin, W. Hadikurniawati, S. Wibisono, H. Februariyanti, and J. S. Wibowo, "A Hybrid Method of Rule-based and String Matching Stemmer for Javanese Language" *J. Theor. Appl. Inf. Technol.*, vol. 15, p. 19, 2017, [Online]. Available: www.jatit.org

[12] M. A. Nq, L. P. Manik, and D. Widiyatmoko, "Stemming Javanese: Another Adaptation of the Nazief-Adriani Algorithm," in *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 627–631. doi: 10.1109/ISRITI51436.2020.9315420.

[13] 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.

[14] N. W. Wardani and P. G. S. C. Nugraha, "Stemming Teks Bahasa Bali dengan Algoritma Enhanced Confix Stripping," *Int. J. Nat. Sci. Eng.*, vol. 4, no. 3, pp. 103–113, Dec. 2020, doi: 10.23887/ijnse.v4i3.30309.

[15] M. Agus, P. Subali, and C. Fatichah, "Kombinasi Metode Rule-based and N-Gram Stemming untuk Mengenali Stemmer Bahasa Bali," vol. 6, no. 2, pp. 219–228, 2019, doi: 10.25126/jtiik.201961105.

[16] J. Elektronik, I. K. Udayana, I. Gede, A. P. Arimbawa, N. Agus, and S. Er, "Lemmatization in Balinese Language". *Jurnal Elektronik Ilmu Komputer Udayana p-ISSN* 2301: 5373, 2017.

[17] I.P.M. Wirayasa, I.M.A. Wirawan, and I.M.A. Pradnyana, "Algoritma Bastal: Adaptasi Algoritma Nazief & Adriani Untuk Stemming Teks Bahasa Bali," *Jurnal Nasional Pendidikan Teknik Informatika: JANAPATI*, *8*(1), pp.60-69.

[18] P. Gede Surya Cipta Nugraha and N. Wayan Wardani, "Stemming Dokumen Teks Bahasa Bali Dengan Metode Rule Base Approach," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 7, no. 3, pp. 510-521, 2020.

[19] F. H. Rachman, N. Ifada, S. Wahyuni, G. D. Ramadani, and A. Pawitra, "ModifiedECS (mECS) Algorithm for Madurese-Indonesian Rule-Based Machine Translation," in *2022 International Conference of Science and Information Technology in Smart Administration, ICSINTESA 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 51–56. doi: 10.1109/ICSINTESA56431.2022.10041470.

[20] E. Lindrawati, E. Utami, and A. Yaqin, "Comparison of Modified Nazief&Adriani and Modified Enhanced Confix Stripping algorithms for Madurese Language Stemming," *INTENSIF J. Ilm. Penelit. dan Penerapan Teknol. Sist. Inf.*, vol. 7, no. 2, pp. 276–289, Aug. 2023, doi: 10.29407/intensif.v7i2.20103.

[21] Enni Lindrawati, Ema Utami, and A. Yaqin, "ANoM STEMMER: Nazief & Andriani Modification for Madurese Stemming," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 6, pp. 1341–1347, Dec. 2023, doi: 10.29207/resti.v7i6.5086.

[22] I. Setiawan and H. Y. Kao, "SUSTEM: An Improved Rule-based Sundanese Stemmer," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 23, no. 6, Jun. 2024, doi: 10.1145/3656342.

[23] A. Ardiyanti Suryani, D. Hendratmo Widyantoro, A. Purwarianti, and Y. Sudaryat, "The rule-based sundanese stemmer," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 17, no. 4, Jul. 2018, doi: 10.1145/3195634.

[24] A. Maesya, Y. Arifin, A. Zahra, and W. Budiharto, "Development of Sundanese Stemmer Based on Morphophonemics," in *10th International Conference on ICT for Smart Society, ICISS 2023 - Proceeding*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICISS59129.2023.10291840.

[25] A. Sutedi, R. Elsen, and M. R. Nasrulloh, "Sundanese Stemming using Syllable Pattern," *J. Online Inform.*, vol. 6, no. 2, p. 218, Dec. 2021, doi: 10.15575/join.v6i2.812.

[26] S. H. Wibowo and S. Wibowo, "Development of Stemming Algorithm for Rejang Language Stemmer Based on Rejang Language Morphology View project Development of Stemming Algorithm for Rejang Language Stemmer Based on Rejang Language Morphology," *Artic. J. Adv. Res. Dyn. Control Syst.*, vol. 11, 2019, [Online]. Available: https://www.researchgate.net/publication/341307354

[27] S. H. Wibowo, R. Toyib, M. Muntahanah, and Y. Darnita, "Time complexity in rejang language stemming," *J. INFOTEL*, vol. 14, no. 3, pp. 174–179, Aug. 2022, doi: 10.20895/infotel.v14i3.764.

[28] R. Sovia, S. Defit, Yuhandri, and Sulastri, "Development of natural language processing on morphology-based Minangkabau language stemming algorithm," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 31, no. 1, pp. 542–552, Jul. 2023, doi: 10.11591/ijeecs.v31.i1.pp542-552.

[29] R. Sovia, S. Defit, and Yuhandri, "Development of the Minangkabau Local Language Translation Machine Based on Stemming," in *Proceeding - 2022 International Symposium on Information Technology and Digital Innovation: Technology Innovation During Pandemic, ISITDI 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 195–198. doi: 10.1109/ISITDI55734.2022.9944457.

[30] Yusra, M. Fikry, and Hendi, "Stemmer bahasa melayu riau berdasarkan aturan morfologi." In *Seminar Nasional Teknologi Informasi Komunikasi dan Industri*, 2021, pp. 118-124.

[31] Z. Abidin, A. Junaidi, Wamiliana, F. R. Lumbanraja, D. Kurniasari, and R. I. Borman, "Rule-Based Dialect of Tulang Bawang Stemmer," in *2025 International Conference on Advancement in Data Science, E-learning and Information System (ICADEIS)*, IEEE, Feb. 2025, pp. 1–6. doi: 10.1109/ICADEIS65852.2025.10933405.

[32] Z. Abidin, A. Wijaya, and D. Pasha, "Aplikasi Stemming Kata Bahasa Lampung Dialek Api Menggunakan Pendekatan Brute-Force dan Pemograman C#," *J. MEDIA Inform. BUDIDARMA*, vol. 5, no. 1, p. 1, Jan. 2021, doi: 10.30865/mib.v5i1.2483.

[33] A. Guterres, Gunawan, and J. Santoso, "Stemming Bahasa Tetun Menggunakan Pendekatan Rule Based," *Teknika*, vol. 8, no. 2, pp. 142–147, Oct. 2019, doi: 10.34148/teknika.v8i2.224.

[34] Z. Abidin, A. Junaidi, and Wamiliana, "Text Stemming and Lemmatization of Regional Languages in Indonesia: A Systematic Literature Review," *J. Inf. Syst. Eng. Bus. Intell.*, vol. 10, no. 2, pp. 217–231, Jun. 2024, doi: 10.20473/jisebi.10.2.217-231.

[35] A. Maesya, A. Ramadhan, E. Abdurachman, A. Trisetyarso, and M. Zarlis, "Stemming Algorithm for the Indonesian Language: A Scientometric View," in *2022 IEEE Creative Communication and Innovative Technology, ICCIT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICCIT55355.2022.10119050.