Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 24, No. 2, March 2025, pp. 321∼332
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 200/M/KPT/2020
DOI: 10.30812/matrik.v24i2.4833

Evaluation Analysis of the Necessity of Stemming and Lemmatization in Text Classification

Ni Wayan Sumartini Saraswati[1], Christina Purnama Yanti[1], I Dewa Made Krishna Muku[1], Dewa Ayu Putu Rasmika Dewi[2]
[1]Institut Bisnis dan Teknologi Indonesia, Denpasar, Indonesia
[2]Monash University, Melbourne, Australia

Article Info

Article history:

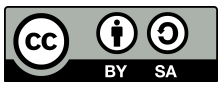
Received January 21, 2025
Revised February 27, 2025
Accepted March 10, 2025

Keywords:

Lemmatization;
Performance;
Stemming;
Support Vector Machine;
Text Classification.

ABSTRACT

Stemming and lemmatization are text preprocessing methods that aim to convert words into their root and to the canonical or dictionary form. Some previous studies state that using stemming and lemmatization worsens the performance of text classification models. However, some other studies report the positive impact of using stemming and lemmatization in supporting the performance of text classification models. This study aims to analyze the impact of stemming and lemmatization in text classification work using the support vector machine method, in this case, devoted to English text datasets and Indonesian text datasets, and analyze when this method should be used. The analysis of the experimental results shows that the use of stemming will generally degrade the performance of the text classification model, especially on large and unbalanced datasets. The research process consisted of several stages: text preprocessing using stemming and lemmatization, feature extraction with Term Frequency-Inverse Document Frequency (TF-IDF), classification using SVM, and model evaluation with 4 experiment scenarios. Stemming performed the best computation time, completing in 4 hours, 51 minutes, and 41.3 seconds on the largest dataset. While lemmatization positively impacts classification performance on small datasets, achieving 91.075% accuracy results in the worst computation time, especially for large datasets, which take 5 hours, 10 minutes, and 25.2 seconds. The Experimental results also show that stemming from the Indonesian balanced dataset yields a better text classification model performance, reaching 82.080% accuracy.



Corresponding Author:

Ni Wayan Sumartini Saraswati, +62 812 4653 960,
Faculty of Technology and Informatics,
Institut Bisnis dan Teknologi Indonesia, Denpasar, Indonesia,
Email: sumartini.saraswati@instiki.ac.id.

How to Cite:
N. W. Saraswati, C. P. Yanti, I. D. M. K. Muku, and D. A. Dewi, "Evaluation Analysis of the Necessity of Stemming and Lemmatization in Text Classification ", *MATRIK: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, Vol.24, No.2, pp. 321-332, March, 2025.

1. INTRODUCTION

Natural Language Processing (NLP) constitutes a multifaceted field situated at the intersection of linguistics, computer science, and artificial intelligence. Its primary objective is to facilitate seamless computer-human interaction by enabling computers to process and analyze substantial volumes of human language data effectively [1]. Stemming emerges as a significant preprocessing technique within NLP. This technique reduces words to their base form, the stem, by systematically removing prefixes, suffixes, and inflectional endings. Acknowledging that the resulting stem may not always correspond to a lexically valid word within the language is crucial. Nevertheless, it is a concise and simplified representation encompassing all related word forms.

The benefits of stemming are that it reduces redundancy in text data by mapping similar words to the same root word. Secondly, it simplifies the representation of text, making NLP tasks more efficient. Stemming is very important in NLP, as this procedure facilitates the analysis of textual data and improves the accuracy of classification and information retrieval [1]. By reducing words to their root form, stemming facilitates the consolidation of related words, improves search accuracy, and reduces redundancy [1]. Some other benefits include better information retrieval, more accurate text classification, less storage space usage, easier sentiment analysis, normalized text, reduced noise in text data, better information extraction process, and simplified language processing tasks. Despite its great benefits in NLP, stemming has limitations where words can lose meaning where prefixed words can lose their grammatical or contextual meaning (e.g., ”relational” becomes ”relat”). Excessive word division, where aggressive truncation can cause unrelated words to be mapped to the same root word (e.g., ”universe” and ”university”). Under-wording, where insufficient truncation can fail to group related words (e.g., ”automate” and ”automation”). Overcoming some of these problems, some NLP research uses lemmatization as a substitute for stemming. Lemmatization is an NLP processing technique that reduces words to their canonical or dictionary form, known as lemmas, while preserving their meaning and grammatical context. Unlike stemming, which simply truncates words to their root form using rule-based heuristics, lemmatization considers the part of speech (POS) and the word's meaning. One of the important jobs in NLP is text classification, which is a fundamental task in NLP [2]. Text classification is the process of categorizing or labeling text data using established criteria. Text classification is commonly utilized in various applications, including spam detection, sentiment analysis, hate speech identification, subject categorization, and many more. Classifying text gets more difficult when created from varied sources such as business, social media, education, and e-commerce [3].

The study in [4] explored the impact of stemming in text classification using lexical features. This research utilizes 30 English electronic essays focusing on several topics, including politics, history, science, prose, sports, and food. The classification methods used are linear hierarchical clustering and non-linear clustering (SOM) for both stemmed and non-stemmed data. Another clustering case conducted by [5] focused on Ukrainian-language tweets. This study compared three text representation techniques: Term Frequency-Inverse Document Frequency (TF-IDF), Bag of Words (BoW), and Bidirectional Encoder Representations from Transformers (BERT). In addition, they analyzed the impact of lemmatization and stemming on clustering quality. The clustering methods used in this study included K-Means, Agglomerative Hierarchical Clustering, and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). This study evaluated various combinations of these clustering techniques with different text processing approaches, TF-IDF, BOW, and BERT, as well as three-word processing scenarios: lemmatization, stemming, and original (without stemming or lemmatization). Another study [6] applied various combinations of 12 preprocessing techniques to three English Twitter datasets related to hate speech and abusive language to improve the quality of short texts. Five machine learning methods and two deep learning methods were used for the classification: SVM, Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, RNN, and CNN. Research involving other languages, such as Uyghur, Kazakh, and Kirghiz, was conducted by [7], who proposed a stemming method across multiple languages to study morpho-phonetic changes based on character-based embeddings and sequential modeling. Research [8] utilizes Twitter data to examine the impact of preprocessing techniques by implementing six preprocessing techniques with various machine learning models as classifiers. Despite these studies, no research has specifically used SVM as a classifier to compare stemming and lemmatization for Indonesian and English languages. This study aims to fill this gap by using review data from Booking.com and Twitter related to hate speech. Additionally, this study examines the impact of preprocessing on both large and small datasets to analyze the effect of preprocessing across different data scales.

More research is needed to investigate the impact of preprocessing on pre-trained models in natural language processing tasks like text classification [9]. The results [4] show that the effect of stemming light on the accuracy of topic-based text classification is ineffective. Accuracy neither increases nor decreases in stemming text. The use of stemming and lemmatization also decreased the quality of the classification model [9]. The findings of [5] revealed that stemming can reduce the computation time, as can lemmatization. However, while faster lemmatization can speed up the process, it also negatively affects model accuracy. However, choosing the best combination of preprocessing techniques can significantly improve classification accuracy [6]. The results of the opposite research state that using stemming and lemmatization in text classification increases the model's ability. The study also showed that stemming produced the best results from the CNN model [10]. Using different machine learning models, such as Naive Bayes, Random Forest, and BiLSTM, stemming and lemmatization also improved the accuracy of the text classification model in the

study [7, 8]. Referring to the differences in the results of these studies, there is a curiosity to examine the impact of stemming and lemmatization on text classification. In more detail, this study aims to determine the impact of using stemming and lemmatization in text classification work, especially in English sentiment analysis and hate speech detection in Indonesian, and analyze when this is needed to obtain the purpose of implementing text classification. It is hoped that with this research, future researchers can consider stemming and lemmatization better instead of just following to use it because most of the previous researchers also used it in text classification work.

2. RESEARCH METHOD

This research uses two datasets, namely English datasets and Indonesian datasets. The English dataset is Hotel Reviews in Europe data taken from the Kaggle website [11]. This review data comes from user reviews on 1493 luxury hotels in Europe sourced from Booking.com. This dataset consists of 387.848 rows for negative sentiment and 479.792 for positive sentiment. The Indonesian dataset is taken from research that has been annotated and consulted with linguistic experts This dataset is a Tweet from social media Twitter that contains hate speech and abusive language [12]. There are 13 columns consisting of hate speech labels, abusive language labels, and the level and category of hate speech. In this research, only hate speech labels will be used. The dataset consists of 13.169 Tweets. A total of 7.608 Tweet rows show non-hate speech, and 5.561 that show hate speech. The flow of text classification experiments in this study consists of several stages, as shown in Figure 1. After obtaining the dataset, the data-cleaning process is carried out. Raw data is not always clean and contains noise, data inconsistencies, and missing values [13]. This data-cleaning process is done to improve data quality [14]. Machine learning models are said to be of high quality if the input data to be trained is also of high quality [15]. The cleaning process includes checking for duplicate data, and missing values, and removing irrelevant columns [16–19].

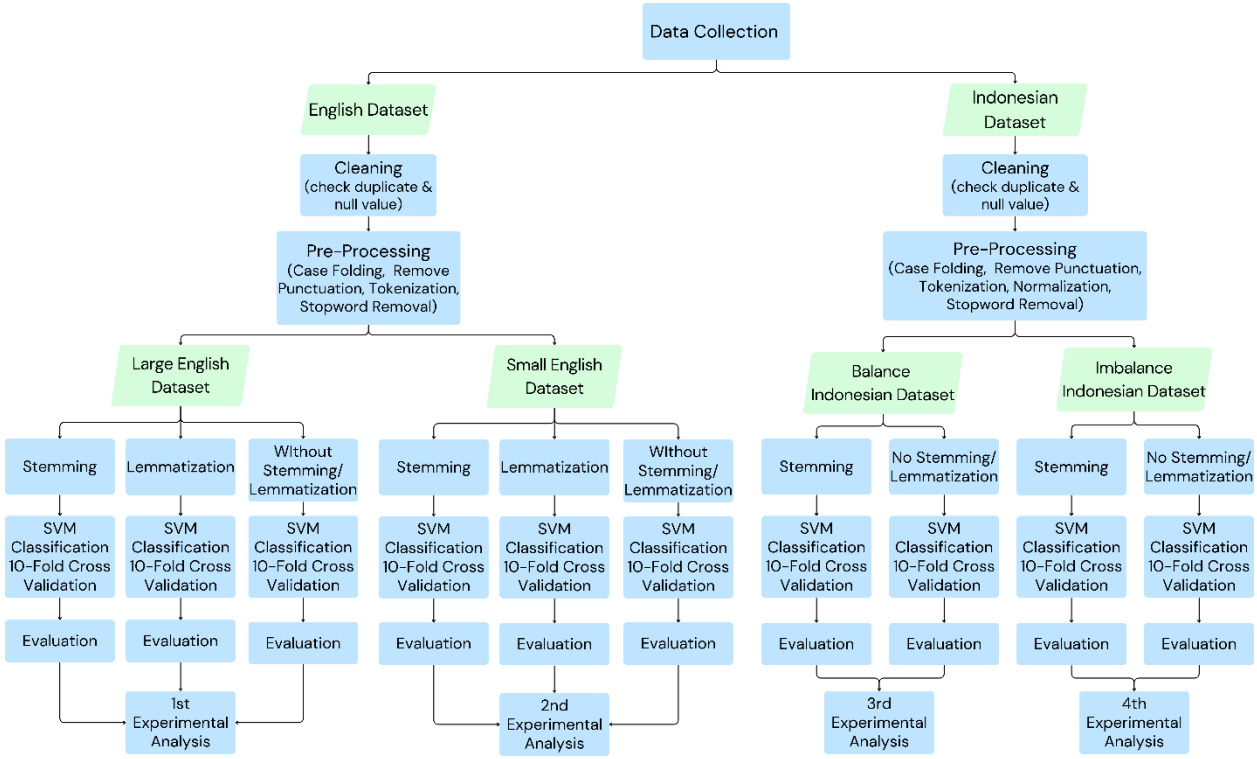
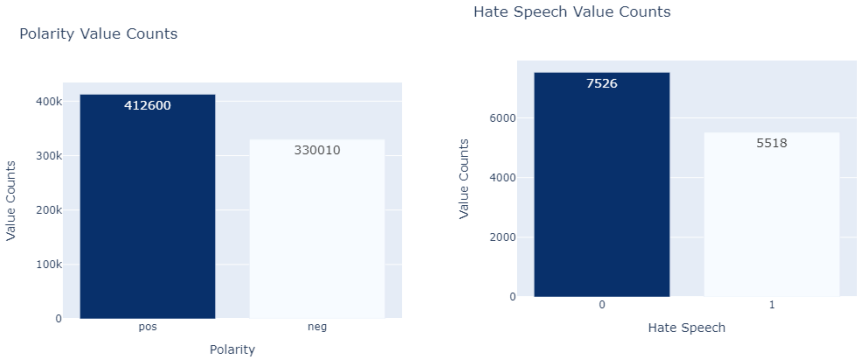


Figure 1. Experimental design

After cleaning the data, we also checked the number of each label in the dataset. The number of labels for the English dataset and the Indonesian dataset after the cleaning stage is shown in the graph in Figure 2 below. In Figure 2(b), label 1 represents the class of tweets that contain hate speech, while label 0 represents the class of tweets that do not contain hate speech.



(a) (b)

Figure 2. Number of labels (a) English dataset, (b) Indonesian dataset

The next step after data cleaning is text preprocessing. Text pre-processing has several stages, namely case folding, removing punctuation, normalization (for Indonesian datasets), and word removal. Case folding in text preprocessing converts data into uniform or lowercase. This process aims to ensure data consistency so that text containing the words ”Good” and ”good” will not be treated differently. This will also have an impact on data accuracy. After folding the case, the text will go through the process of removing punctuation. Text contains words and sentences, punctuation marks, emoticons, symbols, URLs, special characters, and HTML tags. This can add noise to the text. This stage can ensure that the analysis focuses on the semantic content of the text, because punctuation does not contain semantic information [20]. Tokenization is the process of breaking down text into tokens after removing punctuation. It is important to make it easier for machine learning models to process text.

In Indonesian texts, especially Twitter, many users use informal language. This research performs normalization using a slang dictionary obtained from GitHub. There are several slang dictionaries along with the formal language. This is done to clarify meaning and standardize variations. Slang, especially in Indonesian, has many variations, such as the words ”gak,” ”ga,” and ”ngga,” which have the same meaning, namely ”not.” The stop word removal stage is a process to remove words that are irrelevant or do not have semantic meaning [21]. Stop words have no additional information so that they will cause noise. In addition, each word in the text is a feature, so when the stop word removal process is carried out, it can reduce the feature dimension and prevent the model computation from becoming heavier. This research utilizes the stop words function from the NLTK library. For Indonesian data, use the Indonesian stop words list; for English data, use the English stop words list. The English dataset will be truncated in this research to save computation time. This research will use 150.000 review data for each positive review and negative review label. The amount of data used is summarized in the experimental scenario, as shown in Table 1 below. Based on the experiment scenario table, the first experiment will implement the model for a large English dataset with a size of 300.000 reviews. The second experiment still implements English data but with a much smaller dataset size of 5.000 reviews. The third experiment implemented an Indonesian dataset for balanced data totaling 11.036 sentences. The fourth experiment implemented an Indonesian dataset with an unbalanced amount of data. Furthermore, each dataset will go through a different process, and the English dataset will go through lemmatization, stemming, and without stemming/lemmatization, while the Indonesian dataset will go through stemming and without stemming/lemmatization.

Table 1. Experiment Scenario

Scenario	Language	Number of Datasets	Text Pre-processing
1	English	300.000 datasets: 150.000 negative, 150.000 positive	Lemmatization Stemming Without Lemmatization or Stemming
2	English	5.000 datasets: 2.500 negative, 2.500 positive	Lemmatization Stemming Without Lemmatization or Stemming
3	Indonesian	11.036 datasets: 5.518 data of hate speech, 5.518 data not hate speech	Without Lemmatization or Stemming Stemming
4	Indonesian	13.044 datasets: 5.518 data of hate speech, 7.526 data not hate speech	Without Lemmatization or Stemming Stemming

Stemming and lemmatization are two processes with the same goal: reducing words to their basic form, but they have different approaches. Lemmatization converts words into their basic form. The base form or root form of the word in lemmatization is called a lemma [22]. This is in contrast to stemming, which directly cuts the suffix on the word to get the base word (stem). Stemming does not pay attention to the meaning of the word because it directly cuts the suffix and some morphological rules, so there is ambiguity and uncertainty [7]. For example, ”runner” becomes ”run,” or ”studies” becomes ”study,” compared to lemmatization, which changes to the root word; for example, ”running” becomes ”run,” and ”studied” becomes ”study.” Indonesian data uses stemming by utilizing the Sastrawi library, a library for stemming in Indonesian. For English data, stemming and lemmatization are performed using the NLTK library, namely PorterStemmer for stemming and WordNetLemmatizer for lemmatization. The results of data that has been lemmatized, stemmed, or treated will enter the weighting stage. This stage converts text data into vector form before entering SVM modeling. The weighting technique that this research uses is TF-IDF. TF calculates the frequency of words (term t) that often appear in documents (d), and then for IDF, it measures the importance of a word namely, the less often the word appears, the higher the IDF value [23]. The result of weighting is obtained from the product of TF and IDF [23]. In Equation 1, ft, d is the number of occurrences of t in d, and Nd is the total number of words in the document. |D| is the total number of documents in corpus D, and df(t) is the number of documents in corpus D that contain the term t. Adding 1 in Equation 2 aims to avoid division by 0 when no term appears in any document. The feature weight value used in the vector is calculated using Equation 3.

$$TF(t,d) = \frac{f_{t,d}}{N_d} \tag{1}$$

$$IDF(t,d) = log\left(\frac{|D|}{1+df(t)}\right) \tag{2}$$

$$TFIDF(t,d,D) = TF(t,d).IDF(t,D) \tag{3}$$

SVM is one of the supervised learning methods based on Structural Risk Minimization (SRM) [24], which balances the suitability of training data with the complexity of the model. SVM has several advantages, such as good generalization and working for high-dimensional data. SVM can handle both classification and regression cases [25]. In addition, SVM can also be used for linear and non-linear data. SVM works by finding a hyperplane (separating function) to separate observations that have different target values with a maximum margin represented by Equation 4, which is the largest distance between the hyperplane and support vectors (the closest data from each class) [20]. This support vector determines the position of the hyperplane, which represents each class in the classification as represented by equations 5 and 6. Figure 3 illustrates the concept of how the SVM works, where the slanted separating line represents the hyperplane, the distance between the slanted line and the dashed lines represents the margin, and the support vectors are the points located on the dashed lines. Margin is measured by the following Equation 4 [26]:

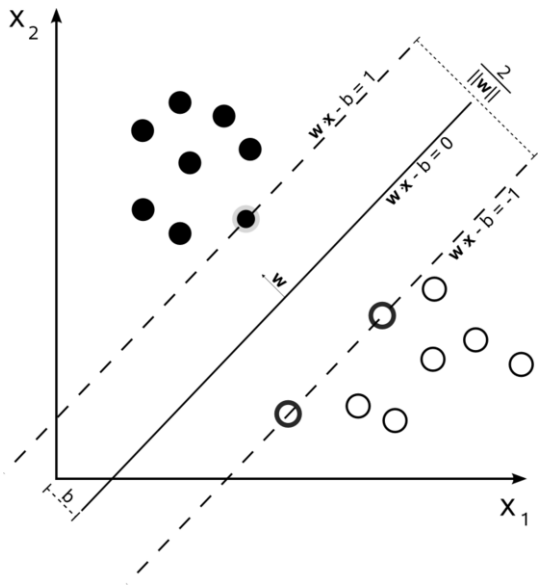


Figure 3. Maximal margin hyperplane

$$\frac{2}{||w||} \tag{4}$$

Equation 4 defines w is the weight vector that determines the direction of the separating hyperplane. The term |w| represents the norm of the weight vector, indicating the magnitude of its contribution to the margin separation. This equation describes the maximum margin width between two classes. A larger margin implies that SVM can distinguish between the two classes more effectively. For lines to separate the data, certain rules must be followed [26]:

$$(w^T x + b) \geq +1 \tag{5}$$

$$(w^T x + b) \leq -1 \tag{6}$$

Equation 5 represents class +1, meaning a data point in the positive class must satisfy this equation. Similarly, Equation 5 represents class -1; therefore, a data point in the negative class must satisfy it. In these equations, x represents the data point, and b is the bias that shifts the hyperplane position. The term w^T denotes the transpose of w, which means that if w is a column vector, w^T is a row vector. These equations help to define the decision boundary for classification.

The classification model that has been built needs to be evaluated. This research evaluates the model's accuracy using several metrics: accuracy, precision, recall, f1-score, and specificity. Accuracy shows the percentage of the model predicting correctly from the total data. Precision shows how well the model predicts true positives from the total predicted positives [27]. Precision can describe how well the model predicts positives for text data without incorrectly predicting negative classes as positives. Recall shows how well the model correctly predicts positive classes. Recall identifies the number of positive classes that the model finds. While specificity shows how well the model predicts negative classes [28]. The F1 score shows the average balance between recall and precision. If one of the metrics between precision and recall is of poor quality, then the F1-score will also be of poor quality. This research uses cross-validation, one of the model testing techniques, to get more stable estimates and avoid bias. Cross-validation divides the test and training data into several splits with the parameter k. This research uses K=10, which means the model is trained 10 times with the proportion of training data and test data being 80%: 20%. The evaluation results will then be averaged so that this technique can generalize to data that is not visible during training.

3. RESULT AND ANALYSIS

The experimental results for English and Indonesian texts are shown in Table 2 below. The experimental results show that the highest accuracy for English data in the first scenario is the group with the treatment without stemming/lemmatization, reaching 93.166%, followed by lemmatization, reaching 93.162%, and stemming has the lowest accuracy, reaching 92.930%. In the second scenario, treatment with lemmatization has the highest accuracy, reaching 91.075%, followed by without lemmatization/stemming, reaching 90.900%, and stemming has the lowest accuracy, reaching 90.850%. Indonesian experiments with balanced data show that data that goes through the stemming process using literary has a higher accuracy, reaching 82.080%, compared to the treatment without stemming/lemmatization, reaching 81.887%. On imbalanced data, higher accuracy is shown by the treatment without stemming/lemmatization, reaching 82.626% compared to stemming with Sastrawi, reaching 81.926%.

Table 2. Model Testing Results

Scenario	Language	Text Pre-processing	Accuracy	Precision	Recall	Specificity	F1-score
1	English	Lemmatization	93.162%	93.192%	93.162%	93.161%	93.160%
		Stemming	92.930%	92.959%	92.930%	92.928%	92.928%
		Without Lemmatization/Stemming	93.166%	93.195%	93.166%	93.165%	93.165%
2	English	Lemmatization	91.075%	91.217%	91.075%	91.079%	91.069%
		Stemming	90.850%	90.977%	90.850%	90.860%	90.845%
		Without Lemmatization/Stemming	90.900%	91.033%	90.900%	90.905%	90.894%
3	Indonesian	Stemming Balance Data	82.080%	82.127%	82.080%	82.070%	82.073%
		Without Lemmatization/Stemming Balance Data	81.887%	81.938%	81.887%	81.872%	81.878%
4	Indonesian	Stemming Imbalance Data	81.926%	81.933%	81.926%	80.946%	81.799%
		Without Lemmatization/Stemming Imbalance Data	82.626%	82.659%	82.626%	81.638%	82.495%

Why English stemming gives the worst results can be due to several reasons, including loss of context, i.e., stemming can sometimes oversimplify, leading to a loss of nuance. For example, "better" can be incorrectly stemmed into "bet," Language-Specific

Rules, where the stemming algorithm may not work well for all languages, and Accuracy Tradeoff, where aggressive stemming can group words with different meanings under the same root word, thus affecting classification accuracy.

Why lemmatization performs worse on large datasets can be explained as follows. In large datasets, different word forms (such as run, running, ran) usually occur quite frequently, allowing the model to learn the relationship between these word forms. Lemmatization unifies all word forms into a root (lemma), reducing this variation, so the model loses features that are actually informative. Example: "The system is running smoothly" and 'The system ran smoothly' will be simplified to the same form, despite the different time context and nuances where the first sentence can be positive and the second sentence can be negative. In small datasets, lemmatization helps by reducing sparsity in the data. However, in large datasets, the variety of word forms is usually represented quite well; sufficient word frequency helps the model understand the relationship between different word forms without simplifying them. As a result, lemmatization no longer provides significant benefits. In small datasets, word form variation (e.g., run, running, ran) can cause the distribution of features to be sparse, which makes it difficult for the model to find relevant patterns. Lemmatization unifies these variations into a basic form (run), reduces the number of unique features, and helps the model work more effectively with limited data. Lemmatization is more useful for small datasets as it helps reduce sparsity, increase generalization, and maximize the available information. In large datasets, the variety of word forms is usually already adequately represented, so lemmatization provides less benefit or is even unnecessary. In the third scenario, stemming in Indonesia with balanced data gets higher evaluation results than without stemming/lemmatization for several reasons. Although both are stemming techniques, stemming specifically designed for Indonesian text cuts not only suffixes but also prefixes, infixes, suffixes, and combined affixes.

Given that Indonesian has a more complex morphology, a variety of vocabulary, and a level of ambiguity, it has challenges different from English. The feature dimension when using stemming is reduced, thus reducing model complexity and redundancy. Stemming also reduces sparsity in text representation to make it denser and more informative. Indonesian still has few annotation datasets, so stemming helps maximize information by ensuring that each word is effectively represented. Removing affixes in Indonesian datasets can reduce noise in the data. In text classification, stemming reduces overfitting; i.e., in a small corpus, unnecessary word variations can cause the model to focus too much on irrelevant details. Here, stemming helps simplify the data to make the model more robust. Indonesian words with similar meanings but different forms are mapped to the same root word. This leads to a more consistent text representation for the classification model. Better generalization of the model to unseen data can occur with stemming, as it maps the different word forms of words to the root word, allowing the model to recognize patterns more effectively. The imbalanced data shows that the dataset without stemming/lemmatization has a higher classification performance than the dataset using stemming. This is due to several reasons. In imbalanced datasets, minority classes often have specific and unique word or phrase patterns. Stemming simplifies words to their base form and can remove these nuances, resulting in important features representing the minority class being underrepresented. Stemming also often removes word variations that are important for recognizing the specific patterns of minority classes. This can exacerbate the imbalance as the minority class loses its distinctive features that help the model distinguish them from the majority. As a result, the model loses the ability to recognize specific patterns in the minority data, which is important for correcting imbalance. The model has more features to learn by retaining the original word form. The model's unit of measurement also reinforces this. If based on specificity, both stemming and without stemming/lemmatization are less able to recognize negative or minority classes. In addition, the higher F1- score shows that the balance between precision and recall is better. This indicates that recall and precision are higher because the model focuses on recognizing more majority classes.

A comparison between the results of this study and similar previous research, such as [4], indicates that the presence or absence of stemming does not significantly impact classification performance. The results remained consistent, showing neither an increase nor a decrease, regardless of whether stemming was applied. There was a high level of consistency between the six main groups identified by hierarchical clustering analysis and those identified through SOM. Despite minor differences, the results obtained from hierarchical clustering were aligned with those obtained from the SOM. In [4], a light stemmer (UEA stemmer) was used, which preserves lexical meaning after stemming. In contrast, this study found that stemming resulted in lower accuracy in both the first (92.930%) and second scenarios (90.850%). This discrepancy may be attributed to the use of PorterStemmer, which applies a more aggressive stemming approach that reduces words to their root forms. Meanwhile, [6] categorized the classification results based on the best and worst preprocessing combinations across three datasets: 25.112 (David et al. dataset), 19.968 (Golbeck et al. dataset), and 15.844 (Wassem et al. dataset). The best-performing preprocessing techniques, measured by the F1-score, were lemmatization and lowercasing of words. Among them, lemmatization achieved the highest evaluation scores in most classification cases, particularly on the David et al. dataset, with the highest F1-score of 0.671 using CNN classification. These findings align with this study's, where lemmatization outperformed stemming in the second scenario, especially for smaller datasets achieving an F1-score of 91.069%.

Based on the computation time of the model, as shown in Table 3, the model with stemming provides the best computation time performance of the four experimental scenarios. In the case of large English datasets, using lemmatization gives worse computation time performance than datasets without stemming or lemmatization. However, on smaller datasets, lemmatization gives better

computation time performance.

Table 3. Model Computation Time

Scenario	Language	Text Pre-processing	Model Computation Time
1	English	Lemmatization	5 hours 10 minutes 25.2 seconds
		Stemming	4 hours, 51 minutes, and 41.3 seconds.
		Without Lemmatization/Stemming	4 hours, 56 minutes, and 8.9 seconds
2	English	Lemmatization	4.7 seconds
		Stemming	4.5 seconds
		Without Lemmatization/Stemming	5 seconds
3	Indonesian	Stemming Balance Data	32.6 seconds
		Without Lemmatization/Stemming Balance Data	34.4 seconds
4	Indonesian	Stemming Imbalance Data	42.6 seconds
		Without Lemmatization/Stemming Imbalance Data	47.3 seconds

Based on the concept of stemming that converts words into their basic form, the dimension of TFIDF features is reduced, so the complexity of the SVM model is also reduced. That is the reason why the computation time of the model using stemming is the most efficient. Why models with lemmatization on large datasets have worse computation time than models without stemming/lemmatization can be explained as follows. TF-IDF vectors are often sparse because most documents only contain a small portion of the overall vocabulary. This sparsity can help speed up computation in SVM implementation. The computation time can be optimized if the feature vector is sparse (many elements are zero). Lemmatization can reduce sparsity by combining similar word forms (e.g., ”running” and ”run”), but the effect on large datasets may not be significant. Although lemmatization can reduce the vocabulary size, this reduction is often small compared to the initial vocabulary size on large datasets. Therefore, the time saved on the SVM model may not be enough to offset the overhead of lemmatization. SVM is designed to work well on sparse data, such as TF-IDF representations. Without lemmatization, the data representation may be sparse but still efficient when processed by the SVM algorithm. Lemmatization, by reducing sparsity, can increase data density, which increases computation time for kernel operations or dot products. The second reason is that lemmatization can potentially increase the term frequency weight in vector data; TF weight in TF-IDF is a numerical component that reflects the frequency of word occurrence in documents. The size of this weight can affect numerical stability when SVM performs optimization, especially if the weight is not normalized. In SVM, a kernel (e.g., linear or RBF) calculates the similarity between data pairs. If the TF-IDF vector has large values or is not well distributed (e.g., without normalization), the kernel computation becomes more expensive as it involves more complex numerical operations. Compared to previous research by [5] the findings of this study indicate that stemming can reduce the computation time. The previous study compared the computation times for the K-Means, Agglomerative Hierarchical Clustering, and HDBSCAN algorithms using various text representation techniques: BOW, TF-IDF, and BERT. By comparing lemmatization, stemming, and the original text (without stemming and lemmatization), it was found that stemming resulted in the shortest computation time, except for the stemming + BERT combination in Agglomerative Hierarchical Clustering. Specifically, the computation times for K-Means with stemming + TF-IDF, stemming + BOW, and stemming + BERT were 6.246 seconds, 8.558 seconds, and 385.74 seconds, respectively. For Agglomerative Hierarchical Clustering, the times were 93.66 seconds, 97.14 seconds, and 472.98 seconds, while for HDBSCAN, the computation times reached 378.89 seconds, 378.09 seconds, and 442.50 seconds, respectively. The findings of this study consistently demonstrate that stemming reduces the model complexity. This is due to the fact that the model learns from a smaller number of features, leading to faster computation times.

4. CONCLUSION

The use of stemming gives worse performance compared to treatment without stemming on English datasets because the aggressiveness of stemming can convert words with different meanings into the same form, causing sentences to lose context and experience accuracy tradeoffs. This is also due to language-specific rules in English. Lemmatization supports the performance of classification models on small data but not on large data. Lemmatization is more useful for small datasets because it helps reduce sparsity, increase generalization, and maximize the available information. The variety of word forms in large datasets is usually adequately represented, so lemmatization provides less or no benefit. In unbalanced datasets, minority classes often have specific and unique patterns of words or phrases. Stemming simplifies words to their base form, which can remove these nuances, resulting in important features representing minority classes being underrepresented. Stemming also often removes word variations that are important for recognizing the specific patterns of the minority class. Thus, stemming gives worse performance to text

classification models on unbalanced datasets. Lemmatization on large datasets has worse computation time than models without stemming/lemmatization because lemmatization can reduce sparsity and potentially increase the weight of term frequency in vector data, which can increase the computational complexity of support vector machine models. Future research could consider using deep learning models, such as BERT, to analyze the impact of lemmatization and stemming on model performance. Additionally, data imbalance handling techniques such as the Synthetic Minority Oversampling Technique (SMOTE) can be applied to improve the model accuracy for minority classes.

5. ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to all parties involved in this research for their valuable contributions and support. We are especially grateful to the reviewers for their insightful comments and suggestions, which have greatly improved the quality and presentation of this manuscript.

6. DECLARATIONS

AUTHOR CONTIBUTION

We thank Author 1 for formulating the problem, defining the research objectives, developing the model, and conducting the analysis in this study. We also extend our gratitude to Author 2 for reviewing previous research. Furthermore, we appreciate Authors 3 and 4 for testing the model in this research.

FUNDING STATEMENT

This research did not receive any funding.

COMPETING INTEREST

There is no competing interest in this research.

REFERENCES

[1] Z. Abidin, A. Junaidi, and Wamiliana, "Text Stemming and Lemmatization of Regional Languages in Indonesia: A Systematic Literature Review," vol. 10, no. 2, pp. 217–231, 2024, https://doi.org/10.20473/jisebi.10.2.217-231. [Online]. Available: https://e-journal.unair.ac.id/JISEBI/article/view/50341

[2] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, "A Survey on Text Classification: From Traditional to Deep Learning," vol. 13, no. 2, pp. 1–41, 2022, https://doi.org/10.1145/3495162. [Online]. Available: https://dl.acm.org/doi/10.1145/3495162

[3] M. M. Rahman, A. I. Shiplu, and Y. Watanobe, "CommentClass: A Robust Ensemble Machine Learning Model for Comment Classification," vol. 17, no. 1, pp. 1–20, 2024, https://doi.org/10.1007/s44196-024-00589-3. [Online]. Available: https://link.springer.com/10.1007/s44196-024-00589-3

[4] R. Ahmed, "Exploring The Impact of Stemming on Text Topic-Based Classification Accuracy," vol. 2, no. 2, pp. 204–224, 2024, https://doi.org/10.61320/jolcc.v2i2.204-224. [Online]. Available: https://jolcc.org/index.php/jolcc/article/view/51

[5] Lviv Polytechnic National University, Lviv, 79013, Ukraine, O. Prokipchuk, V. Vysotska, P. Pukach, V. Lytvyn, D. Uhryn, Y. Ushenko, and Z. Hu, "Intelligent Analysis of Ukrainian-language Tweets for Public Opinion Research based on NLP Methods and Machine Learning Technology," vol. 15, no. 3, pp. 70–93, 2023, https://doi.org/10.5815/ijmecs.2023.03.06. [Online]. Available: http://mecs-press.org/ijmecs/ijmecs-v15-n3/v15n3-6.html

[6] U. Naseem, I. Razzak, and P. W. Eklund, "A survey of pre-processing techniques to improve short-text quality: A case study on hate speech detection on twitter," vol. 80, no. 28–29, pp. 35 239–35 266, 2021, https://doi.org/10.1007/s11042-020-10082-6. [Online]. Available: https://link.springer.com/10.1007/s11042-020-10082-6

[7] G. Imin, M. Ablimit, H. Yilahun, and A. Hamdulla, "A Character String-Based Stemming for Morphologically Derivative Languages," vol. 13, no. 4, pp. 1–16, 2022, https://doi.org/10.3390/info13040170. [Online]. Available: https://www.mdpi.com/2078-2489/13/4/170

[8] J. K. Mursi, P. R. Subramaniam, and I. Govender, "Exploring the Influence of Pre-Processing Techniques in Obtaining Labelled Data from Twitter Data," in *2023 IEEE AFRICON*. IEEE, 2023, pp. 1–6, https://doi.org/10.1109/AFRICON55910.2023.10293408. [Online]. Available: https://ieeexplore.ieee.org/document/10293408/

[9] S. F. Chaerul Haviana, S. Mulyono, and Badie'Ah, "The Effects of Stopwords, Stemming, and Lemmatization on Pre-trained Language Models for Text Classification: A Technical Study," in *2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2023, pp. 521–527.

[10] M. Siino, I. Tinnirello, and M. La Cascia, "Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers," vol. 121, March, pp. 1–19, 2024, https://doi.org/10.1016/j.is.2023.102342. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306437923001783

[11] J. Liu, "515K Hotel Reviews Data in Europe," https://www.kaggle.com/datasets/jiashenliu/515k-hotel-reviews-data-in-europe/data.

[12] M. O. Ibrohim and I. Budi, "Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter," in *Proceedings of the Third Workshop on Abusive Language Online*, S. T. Roberts, J. Tetreault, V. Prabhakaran, and Z. Waseem, Eds. Association for Computational Linguistics, 2019, pp. 46–57, https://doi.org/10.18653/v1/W19-3506. [Online]. Available: https://aclanthology.org/W19-3506/

[13] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," vol. 3, no. 1, pp. 91–99, 2022, https://doi.org/10.1016/j.gltp.2022.04.020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666285X22000565

[14] F. Neutatz, B. Chen, Y. Alkhatib, J. Ye, and Z. Abedjan, "Data Cleaning and AutoML: Would an Optimizer Choose to Clean?" vol. 22, no. 2, pp. 121–130, 2022, https://doi.org/10.1007/s13222-022-00413-2. [Online]. Available: https://link.springer.com/10.1007/s13222-022-00413-2

[15] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang, "CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 13–24, https://doi.org/10.1109/ICDE51399.2021.00009. [Online]. Available: https://ieeexplore.ieee.org/document/9458702/

[16] N. W. S. Saraswati, I. K. G. D. Putra, M. Sudarma, I. M. Sukarsa, C. P. Yanti, and N. K. Tri Juniartini, "Revealing the Potential of Hotel Improvements in Bali Based on Sentiment Analysis and Tourist Characteristics," in *2024 11th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. IEEE, 2024, pp. 722–728, https://doi.org/10.1109/EECSI63442.2024.10776092. [Online]. Available: https://ieeexplore.ieee.org/document/10776092/

[17] N. W. S. Saraswati, I. D. M. K. Muku, I. W. D. Suryawan, D. A. K. Pramita, and I. K. A. Bisena, "Balinese Temple: The Image and Characteristics of Tourists based on Sentiment Analysis," in *2024 IEEE International Symposium on Consumer Technology (ISCT)*. IEEE, 2024, pp. 19–24, https://doi.org/10.1109/ISCT62336.2024.10791104. [Online]. Available: https://ieeexplore.ieee.org/document/10791104/

[18] N. W. S. Saraswati, I. Ketut Gede Darma Putra, M. Sudarma, and I. Made Sukarsa, "Enhance sentiment analysis in big data tourism using hybrid lexicon and active learning support vector machine," vol. 13, no. 5, pp. 3663–3674, 2024.

[19] N. W. S. Saraswati, I. K. G. D. Putra, M. Sudarma, and I. M. Sukarsa, "The Image of Tourist Attraction in Bali Based on Big Data Analytics and Sentiment Analysis," in *2023 International Conference on Smart-Green Technology in Electrical and Information Systems (ICSGTEIS)*. IEEE, 2023, pp. 82–87, https://doi.org/10.1109/ICSGTEIS60500.2023.10424322. [Online]. Available: https://ieeexplore.ieee.org/document/10424322/

[20] C. Xu, P. Coen-Pirani, and X. Jiang, "Empirical Study of Overfitting in Deep Learning for Predicting Breast Cancer Metastasis," vol. 15, no. 7, pp. 1–18, 2023, https://doi.org/10.3390/cancers15071969. [Online]. Available: https://www.mdpi.com/2072-6694/15/7/1969

[21] A. Habberrih and M. Ali Abuzaraida, "Sentiment Analysis of Libyan Dialect Using Machine Learning with Stemming and Stop-words Removal," in *5th International Conference on Communication Engineering and Computer Science (CIC-COCOS'24)*. Cihan University-Erbil, 2024, pp. 259–264, https://doi.org/10.24086/cocos2024/paper.1171. [Online]. Available: https://conferences.cihanuniversity.edu.iq/index.php/COCOS/COCOS24/paper/view/1171

[22] S. Shaukat, M. Asad, and A. Akram, "Developing an Urdu Lemmatizer Using a Dictionary-Based Lookup Approach," vol. 13, no. 8, p. 5103, 2023, https://doi.org/10.3390/app13085103. [Online]. Available: https://www.mdpi.com/2076-3417/13/8/5103

[23] A. R. Lubis, M. K. M. Nasution, O. S. Sitompul, and E. M. Zamzami, "The effect of the TF-IDF algorithm in times series in forecasting word on social media," vol. 22, no. 2, p. 976, 2021, https://doi.org/10.11591/ijeecs.v22.i2.pp976-984. [Online]. Available: http://ijeecs.iaescore.com/index.php/IJEECS/article/view/24885

[24] N. M. Guerrero, J. Aparicio, and D. Valero-Carreras, "Combining Data Envelopment Analysis and Machine Learning," vol. 10, no. 6, p. 909, 2022, https://doi.org/10.3390/math10060909. [Online]. Available: https://www.mdpi.com/2227-7390/10/6/909

[25] J. M. Álvarez Alvarado, J. G. Ríos-Moreno, S. A. Obregón-Biosca, G. Ronquillo-Lomelí, E. Ventura-Ramos, and M. Trejo-Perea, "Hybrid Techniques to Predict Solar Radiation Using Support Vector Machine and Search Optimization Algorithms: A Review," vol. 11, no. 3, pp. 1–17, 2021, https://doi.org/10.3390/app11031044. [Online]. Available: https://www.mdpi.com/2076-3417/11/3/1044

[26] F. Nie, Z. Hao, and R. Wang, "Multi-Class Support Vector Machine with Maximizing Minimum Margin," vol. 38, no. 13, pp. 14 466–14 473, 2024, https://doi.org/10.1609/aaai.v38i13.29361. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29361

[27] R. Wang, J. Zhang, Y. Lu, S. Ren, and J. Huang, "Towards a Reliable Design of Geopolymer Concrete for Green Landscapes: A Comparative Study of Tree-Based and Regression-Based Models," vol. 14, no. 3, p. 615, 2024, https://doi.org/10.3390/buildings14030615. [Online]. Available: https://www.mdpi.com/2075-5309/14/3/615

[28] A. Stanzione, R. Cuocolo, L. Ugga, F. Verde, V. Romeo, A. Brunetti, and S. Maurea, "Oncologic Imaging and Radiomics: A Walkthrough Review of Methodological Challenges," vol. 14, no. 19, p. 4871, 2022, https://doi.org/10.3390/cancers14194871. [Online]. Available: https://www.mdpi.com/2072-6694/14/19/4871

[This page intentionally left blank.]