MOTEC: The Malay Offensive Text Classification using Extra Tree and Language Standardization

Fairuz Amalina[1], Faiz Zaki[1], Hamza H. M. Altarturi[2], Hazim Hanif[3], Nor Badrul Anuar [4,5]*

[1]Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya, 50603 Kuala Lumpur, Malaysia

[2]International Center for Living Aquatic Resource Management, WorldFish, 11960 Pulau Pinang, Malaysia

[3]Department of Software Engineering, Faculty of Computer Science and Information Technology, Universiti Malaya, 50603 Kuala Lumpur, Malaysia

[4]Centre of Research for Cyber Security and Network (CSNET), Faculty of Computer Science and Information Technology, Universiti Malaya, 50603, Kuala Lumpur, Malaysia,

[5]Visiting Professor, Institute of Informatics & Computing in Energy, Universiti Tenaga Nasional, Kajang, Malaysia

Emails: fairuzamalina@siswa.um.edu.my[1] , faizzaki@um.edu.my[1] , h.altarturi@cgiar.org[2], hazimhanif@um.edu.my[3], badrul@um.edu.my[4,5]* (Corresponding Author)

ABSTRACT

Cyberbullying has increased globally, with offensive text contributing significantly. Detecting offensive text in Malay is challenging due to non-standard Malay text, unique social media writing styles, a lack of standardization, and limited resources. This study proposes the Malay Offensive Text Classification (MOTEC) framework to address these challenges. The MOTEC framework incorporates a Malay standardization preprocessing task, utilizing three specialized dictionaries: (a) abbreviations, (b) noisy text, and (c) Malaysian dialects. This approach enhances data quality by converting non-standard text into standardized Malay sentences before classification. For feature extraction, the framework employs Term Frequency-Inverse Document Frequency (TF-IDF). This statistical method evaluates the importance of words in a document relative to a collection of documents, coupled with an Extra Tree classifier for the classification process. Evaluating the MOTEC framework using a private dataset collected from Twitter, this study achieved a classification accuracy of 94%, significantly outperforming other studies, which reported an accuracy of 84%. The MOTEC framework substantially improves the classification of offensive Malay text by enhancing accuracy, reducing execution time, and improving data quality through effective language standardization.

Keywords: Natural Language Processing; Machine Learning, Extra Tree, Offensive Text, Text Classification, Decision Tree, Ensemble Method, Malay Language

1.0 INTRODUCTION

The rapid evolution of the Internet and the proliferation of social media have significantly increased cyberbullying, particularly among teenagers and young adults, who are the most affected demographic. Consequently, the Internet has become a breeding ground for bolder and more confident online behaviors, leading to the emergence of cyberbullying [1]. Social media platforms often facilitate the use of harsh and offensive language as in Malaysia, three out of ten young people have reported being victims of cyberbullying [2]. Offensive text usage, including hate speech, harassment, abusive language, sarcasm, and trolling, contributes to aggressive online behaviors and creates a hostile environment [3]. However, the diversity of writing styles, including abbreviations, unstandardized words, and dialects, complicates the detection of offensive text through automated scripts and leads to many of them being misclassified, making the development of effective detection models challenging.

1.1 Background

Existing methods for offensive text classification face several limitations in handling the linguistic diversity of the Malay language. Also, the non-standardized training datasets, which fail to represent the full range of writing styles, result in biased and less accurate models. Researchers often improve data quality through preprocessing

techniques like stop word removal, tokenization, username and hashtag removal, case folding, and stemming [4-7]. However, standardization of non-standard words and dialects remains relatively understudied, particularly in Malay. For example, studies on sentiment analysis often limit their focus to specific dialects, such as the Sabah dialect, and rely on spellcheckers for noisy language normalization [8]. Similarly, other studies have used normalization and stemming to handle abbreviations and noisy text, but overlooked dialectal variations [6, 9]. The lack of comprehensive linguistic resources, dictionaries, and standardization also contributes to additional challenges.

Moreover, existing research has made an effort to explore some of the challenges in various techniques to address non-standard language, dialects, and code-mixed texts [4, 10-12]. For instance, one study proposed an unsupervised normalization model using rule-based patterns and the SymSpell spelling correction algorithm to handle out-of-vocabulary (OOV) words with repeated letters, but it struggled with slang and unconventional language [10]. Another study focused on Malayalam-English code-mixed YouTube comments using a Hierarchical Attention Network (HAN) with BERT tokenization, but faced issues with dataset inadequacy [12]. Additionally, research on multilingual social media content using deep learning highlighted the complexity of handling code-mixed and script-mixed languages due to limited labeled datasets. A study on a Twitter dataset achieved 76% accuracy with TF-IDF and logistic regression, but failed to address special characters and dialects effectively [6]. These limitations emphasize the need for more robust systems for under-resourced languages and diverse dialectal contexts.

To address these issues, this paper aims to improve the classification accuracy of Malay offensive text by addressing these challenges. The proposed Malay Offensive Text Classification (MOTEC) framework enhances dataset quality and handles unfamiliar, non-standard vocabulary. Specifically designed for Malay text, the framework comprises four main phases: input, preprocessing, feature extraction, classification, and output. During the input phase, this study used a keyword search technique to collect data containing abbreviations and noisy text. Four annotators with linguistic expertise manually labeled the input data to create the Malay Offensive Text (MOFFET) dataset, achieving moderate inter-annotator agreement (Fleiss' Kappa = 0.78) [13]. In the preprocessing phase, this study employed a standardization approach that normalizes noisy text, dialectal variations, and abbreviations based on predefined dictionaries, such as Kamus Dewan Bahasa dan Pustaka [14]. Recognizing the multilingual nature of Malaysia, this study developed seven dialect dictionaries for language standardization. In the feature extraction phase, Term Frequency-Inverse Document Frequency (TF-IDF) was applied to weigh terms based on frequency and relevance. Finally, the Extra Tree classifier was used to classify the text, demonstrating superior performance due to its efficiency and robustness.

To develop and evaluate the MOTEC framework, this study initially utilized 5000 tweets from the MOFFET dataset as input. This study then conducted three main experiments:

(a) Experiment 1: Preprocessing for Malay Offensive Text Detection: This study examined the reliability of the proposed standardization method and other preprocessing tasks tailored for the MOFFET dataset.
(b) Experiment 2: Classifying Malay Offensive Text with Different Algorithms: This study tested six machine learning classifiers: Support Vector Machine (SVM), Random Forest (RF), Bernoulli Naive Bayes (BNB), Extra Tree (ET), and Logistic Regression (LR).
(c) Experiment 3: Baseline Studies Comparison Using the MOFFET Dataset: This study compared the MOTEC framework with three baseline studies from similar domains [4, 5, 9]

This study assessed the scores using appropriate performance metrics such as accuracy, standard deviation, and ROC curve, demonstrating the efficacy of the MOTEC framework in classifying Malay offensive text. With promising results, this study makes several significant contributions to the field of Malay offensive text classification:

(a) This study created a self-collected MOFFET dataset comprising 5000 Malay language tweets that include dialects, abbreviations, noisy words, and emojis.
(b) This study proposed a comprehensive Malay Offensive Text Classification (MOTEC) framework specifically designed to detect offensive text in Malay. A significant contribution of our framework lies in language standardization, which eliminates redundant data storage and ensures data dependencies by normalizing vocabulary and replacing dialectal terms with their standard Malay equivalents based on predefined mappings. Without this standardization, synonymous words would remain separate, creating redundancy.
(c) This study conducted three main experiments, preprocessing, classification, and baseline study—to thoroughly evaluate the MOTEC framework. Our MOTEC framework demonstrated a significant improvement in accuracy, achieving an average increase of 21.84% compared to the selected baseline studies.

The rest of the paper is organized as follows. Section 2 discusses existing offensive text and challenges in Malay offensive text classification, providing a comprehensive background for the study of offensive text. Section 3 elaborates on the proposed method designed explicitly for Malay text classification, detailing each step from data collection and annotation to data preprocessing, feature extraction, classification, and evaluation. Section 4 presents our experimental setup, focusing on preprocessing tasks, and explains the results, providing a thorough

analysis and discussion. Finally, Section 5 concludes the paper by highlighting this study's contributions and suggesting future directions for Malay text classification research.

2.0 RELATED WORKS

Offensive text classification is a well-established area within Natural Language Processing (NLP). Most of the existing research has focused on the English language; however, there is a growing interest in exploring this classification in other languages, such as Malay [12, 15-17]. This section discusses the techniques and challenges associated with offensive text classification in Malay.

2.1 Machine Learning Classification

One of the best-known offensive text classification techniques is using machine learning. This technique is broadly classified into two categories: supervised and unsupervised. In particular, the unsupervised learning approach pointed out by Lee, et al. [18] is used to identify abusive words in documents. This approach relies on using several words for abusive and non-abusive word lists to search for matching words in the document to conclude whether the text is abusive. However, this approach has limitations, and its effectiveness in detecting malicious words is uncertain.

Researchers have also utilized supervised learning approaches to detect offensive text [4, 5, 9, 19, 20]. Standard supervised learning techniques include random forest, support vector machine, and naïve Bayes. However, these approaches can suffer from misclassification if data preprocessing is incomplete, as non-standard words and abbreviations can introduce ambiguity that disrupts data patterns.

In addition to machine learning techniques, offensive text classification can leverage features for classification. Features are the text's characteristics used to train the model. Standard features for classification include bag-of-words [21], word embeddings [22], and n-grams [23]. Researchers also extract sentiment-based, semantic, and unigram features. However, these features can miss the context of words, word embeddings require large datasets, and n-grams can be challenging to manage. Improper handling of these features can introduce potential noise, reducing the model's performance.

2.2 Deep Learning Classification

Deep learning has emerged as a promising approach for offensive text classification. There are improvements identified from the literature:

(a) *Excessive preprocessing and loss of context.* Husain and Uzuner [24] Utilized BERT and LSTM models for offensive text detection, applying minimal preprocessing tasks. They argued that excessive preprocessing might eliminate important linguistic cues. Traditional preprocessing techniques often lead to the loss of word meaning. To address this, Maity, et al. [17] proposed XLCaps, a capsule network model combined with FastText and Bi-GRU, to manage noisy Malay text while preserving important context. This model captures complex word relationships using dynamic routing, enhancing the model's ability to accurately detect abusive and hate speech in Malay social media posts.

(b) *Integrating user behavior and social relationships*. Miao, et al. [25] developed the GF-OLD (Graph Attention Network and Fusion Features for Offensive Language Detection) model, which integrates Graph Attention Networks (GATs) and BERT-based textual features to identify offensive language on social media. These approaches construct social graphs using user historical behavior and social relationships, employing GATs to capture community information. This method has shown significant achievements in offensive language detection.

2.3 Malay Language Challenges

In addition to existing offensive text classification methods, researchers have identified several challenges in this domain. Previous studies highlight challenges such as the use of informal and offensive language, the mingling of foreign and local languages, the omission of vowels, the repetition of characters, the substitution of characters and words, the use of proverbs, and the addition of words at the beginning or end of sentences [6, 8, 10, 11, 16, 26]. These were addressed by either ignoring the text or removing the repetition. These challenges significantly hinder the accurate classification of offensive text, particularly in Malay. Researchers have investigated and improved various aspects of these challenges, such as:

(a) *Informal Language.* The strategy to solve informal languages, such as abbreviations and non-standard words, by normalizing the repeated character into the standard form is aimed at reducing ambiguity and keeping the original meaning [4, 5, 9, 27, 28].

(b) *Multi-dialect usage.* A significant challenge in text classification is to capture the nuances of multiple dialects within the Malay language. Sulaiman, et al. [29] selected twenty words to reflect pronunciation differences

among dialects, while Hijazi, et al. [30] addressed this by focusing specifically on the Sabah dialects, which accurately represent the linguistic features.

(c) *Dataset coverage.* Various researchers have taken steps to refine dataset coverage by including multi-dialects from the northern, eastern, and standard languages. Besides, an author creates synthetic parallel datasets by replacing familiar noisy words with predefined rules. These efforts aim to enhance the resources available for accurate offensive text classification in the Malay language [15, 17, 29].

Despite that, these are the challenges in the Malay language that are visible:

(a) *Data availability for non-English languages is deficient*. Most studies focus on languages like English, with vast resources and public data available. However, this limited availability makes it difficult to apply existing methods to other languages, such as Malay [4, 5, 9, 27, 31].

(b) *Detecting newly invented and unknown words is challenging*. Offensive language constantly evolves, and models that rely on predefined lists of offensive words may miss these new terms [4, 5, 18]. Therefore, to detect such words effectively, models must generalize beyond their training data and adapt to new language variations and contexts.

(c) *Detecting offensive language in multilingual environments is complex*. In such environments, models must perform language identification and language-specific processing. This capability is crucial to detect offensive language across all languages without misinterpretation accurately [27].

3 METHODOLOGY

Offensive text classification involves five key components: input, preprocessing, feature extraction, classification, and output phases. Researchers commonly use these components to classify offensive text [6, 9, 20, 27].

Fig. *1* illustrates the MOTEC framework applied in Malay offensive text classification in this study. This approach stands out by employing a preprocessing task that begins with standardization, cleaning, reduction, and transformation. Before proceeding to the next stage, the preprocessing phase aims to reformat non-standard words into their original forms and eliminate incomplete data. After the preprocessing phase, the feature extraction method reduces the dataset's dimensionality and converts the data into a numerical vector. Finally, the vectorized data is fed into a machine learning classifier to categorize the text as offensive or non-offensive.
Table *1* summarizes each phase in the MOTEC framework. The following sections detail each phase of the methodology.

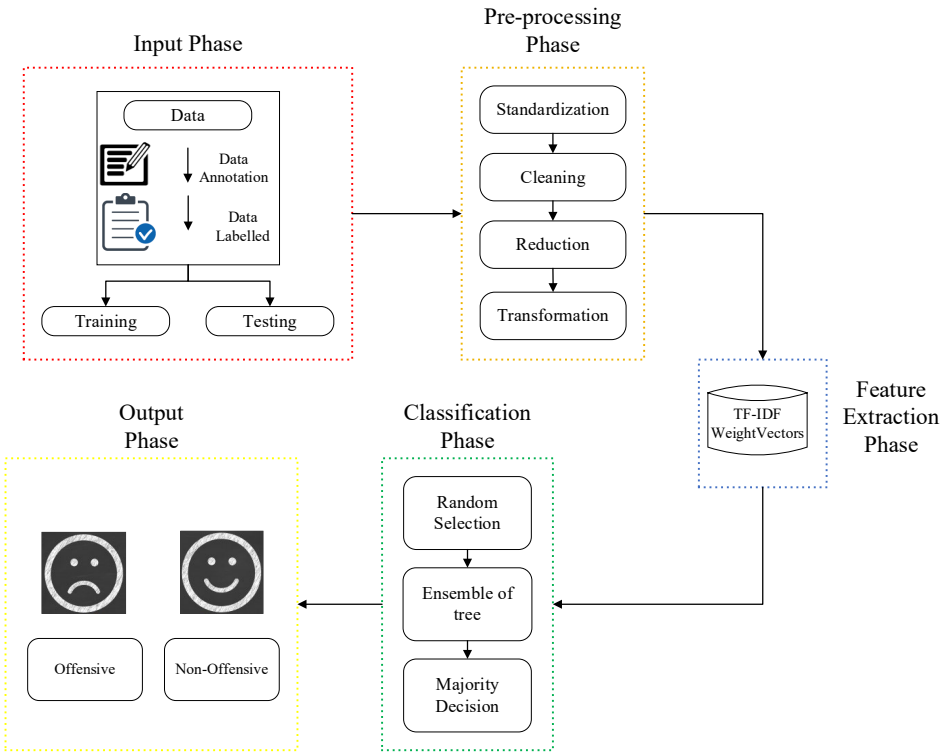

Fig. 1: MOTEC framework

Table 1: MOTEC Framework process

Phases	Input	Process	Output
Input Data	Utilized Twitter API to crawl through the Malay offensive text wordlist by keyword search	362,856 tweets were collected, and manually selected 5000 tweets according to three main criteria. Four annotators annotated the raw data, and the label used was majority voting.	5000 labeled tweets named as MOFFET
Preprocessing	MOFFET dataset	Employed standardization from three dictionaries and a cleaning task for data uniformity	Cleaned and standardized format of the MOFFET dataset version
Feature Extraction	Processed MOFFET dataset	Employed TF-IDF, weight vectors	Features extracted for use in classification
Classification	Extracted features	Random selection, an ensemble of trees, the majority decision	Text classified as Offensive or Non-Offensive

3.1 Input Data

The input data refers to raw data collection and is fed into a model for classification. The objective is to analyze the specific offensive text input data, particularly in Malay. The raw data collection process involved retrieving tweets from Twitter using keyword search techniques over two months. The keyword is using a list of Malay offensive terms that allowed us to collect a substantial volume of 362,856 tweets. Subsequently, a subset of 5,000 Malay tweets was manually selected based on three specific criteria: containing Malay noisy words, dialects, and abbreviations such as in Table 2.

Table 2: Example of non-standard Malay language

Types	Malay Noisy Words	Abbreviations	Dialects
Example	Mantol, palau, weh, lah, kan, je, meh, apsal,	Gitu, skrg, x, jln, mlm, org, bwg, mna, tgk, p, nk, td, brp, blh, tkde, nk,	Demo, ekau, guane, pigi, awok, nau, bior, dapek, mai, habaq, kome, gapo

Next, two annotators with a master's degree and two annotators with a bachelor's degree with Malaysian linguistic variations were hired to annotate the dataset. As a proper guideline, this annotator followed the provided ground truth to label offensive or non-offensive text. Offensive labels are tweets containing vulgar language, targeted insults, or hate speech; meanwhile, non-offensive text is only general comments, not hostile sarcasm or jokes without offensive intent.

The inter-annotator agreement was measured using Fleiss' Kappa, resulting in a value of 0.78, which indicates substantial agreement. This metric highlights the reliability of the annotations for the classification task. Following the annotation process, this study used majority voting for the end label and named the Malay Offensive Text (MOFFET) dataset as an output. The term "dataset" is used because MOFFET is a structured collection of labeled data created specifically for supervised learning in offensive language classification. Unlike a corpus, which is typically an unstructured or general collection of texts, MOFFET is organized with clear labels for this specific task. The Malay MOFFET dataset is divided into 3,847 instances classified as 'non-offensive' and 1,153 instances labeled as 'offensive'.

3.2 Data Processing

Data preprocessing is an essential component in classification, and the objective is to transform raw data into an analyzable format. This study utilized two main preprocessing groups: standardization and cleaning. The sub-sections below elaborate on these preprocessing groups in detail.

Initially, this study subjected the MOFFET dataset as input raw data and went through the standardization process to address non-standard words, abbreviations, and dialects before proceeding to the cleaning task. Notably, non-standard words may contain symbols, such as 'Sh!t' or 'cr@yzy.' It involves three main dictionaries for dialect, noise, and abbreviation to standardize those texts. Then, the cleaning process takes over for data uniformity. After completing the preprocessing steps, the output of processed data serves as the final dataset before entering the feature extraction phase. Fig. 2 details the preprocessing tasks:

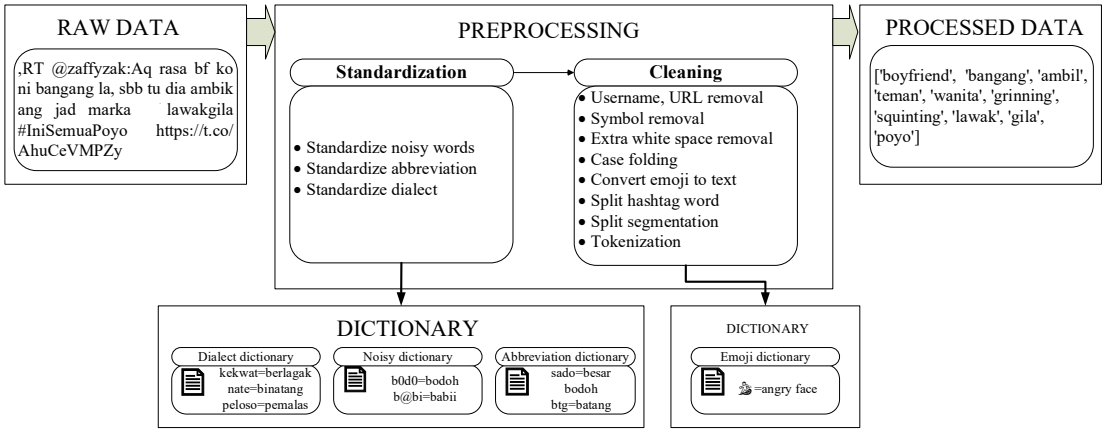

Fig. 2. Preprocessing structure

Table 3 summarizes the data preprocessing task for the MOFFET dataset. It has focused on Standardization and cleaning to improve data quality. Language standardization involves establishing and maintaining the conventional form of language, such as dialects, abbreviations, and noisy language, into a standard form of spelling, text, and language format approved by the country's language council. This process aims to standardize any form of informal language into its standard form. Standardization techniques include standardizing noisy words, abbreviations, and dialects to reduce word redundancy and handle data with varying language forms. Standardizing noisy words, also known as spelling correction, addresses the numerous typos found in real-world texts, whether intentional or unintentional, as demonstrated by Al-Saffar, et al. [26]. The process involves developing dictionaries and using Python functions to detect and correct these issues.

Additionally, standardization expands abbreviations to their original words, which is common in social media writing. The expansion process involves creating an abbreviations dictionary and abbreviations mapped to their full form, replacing these shortened terms in the text. Besides that, Malaysia's diverse dialects, both regional and ethnic, present another challenge. Hijazi, et al. [30] focused on predicting social media sentiment written in the Sabah dialect, highlighting the complexity of standardizing dialects due to their extreme heterogeneity. The insufficient Malay standardization intensifies the heterogeneity of dialectal writing systems.

One of the contributions of this study is the Malay standardization task. This study's unstandardized words include noisy words, abbreviations, and dialects. This study developed a Malay standardization consisting of a noisy language dictionary, an abbreviation dictionary, and a Malay dialect dictionary, covering seven dialects: Kelantan, Pahang, Terengganu, Kedah, Melaka, Negeri Sembilan, and Johor. This study used Python functions for word detection and correction during this phase, employing insertion and replacement methods with the mentioned dictionaries. Importantly, this study chose to maintain the existing unstandardized words rather than remove them, as they contain valuable information that adds meaning to the text. For example, this study replaced the expected informal contraction 'kat' with its standard form 'dekat' (near) and corrected typographic mistakes. This study also standardized dialect-specific words, such as replacing 'kelik' with 'balik' (go back) in standard Malay.

The cleaning focuses on removing irrelevant elements and refining the text. This includes eliminating usernames, URLs, hashtags, symbols, and extra white spaces using regular expressions. Stop words, which add little semantic value, are removed using the NLTK library. Stemming and lemmatization reduce words to their root forms, while emojis are translated into text descriptions using a predefined dictionary. Split segmentation corrects missing spaces for better tokenization, case folding ensures uniformity by converting text to lowercase, and tokenization splits the text into manageable units using the NLTK library. In conclusion, these processes enhance text quality, making it suitable for further analysis and application.

Table 3: Data Preprocessing

Group	Task	Description	Method
Standardization	Noisy words	Addressing typos and incorrect spellings, both intentional and unintentional.	Develop a noisy language dictionary. Python functions are used to detect and correct typos using insertion and replacement methods with the dictionary. 217 Example: Replace 'kat' with 'dekat'.
	Abbreviations	Expanding abbreviations to their full forms.	Create an abbreviation dictionary. Use Python functions to detect and replace abbreviations with their full forms. (1480) Example: Replace 'brb' with 'be right back'.
	Dialects	Standardizing regional and ethnic dialects into standard language.	Develop a Malay dialect dictionary covering dialects such as Kelantan (590 words), Pahang (264 words), Terengganu (155 words), Kedah (311 words), Melaka (382 words), Negeri Sembilan (663 words), and Johor (312 words). Use regular expression functions for word detection and correction. Example: Replace 'kelik' with 'balik'.
Cleaning	Removing Usernames, URLs, Hashtags, Symbols, and Extra White Spaces	Removing irrelevant components from the text to improve data quality.	Use regular expressions to remove usernames, URLs, hashtags, symbols, and extra white spaces. Example: re.sub(r'@\w+', '', text) to remove usernames.
	Removing Stop words	Removing common words that do not add significant meaning to the text.	Use the NLTK library to remove stop words from the text. Example: stop_words = set(stopwords.words('malay')) followed by filtering tokens.
	Stemming	Reducing words to their root or base forms to handle morphological variations.	Use the NLTK library for stemming and lemmatization. Example: Use PorterStemmer for stemming (stemmer.stem(word)) and WordNetLemmatizer for lemmatization (lemmatizer.lemmatize(word)).
	Converting Emojis to Text	Preserving information conveyed by emojis by converting them to text descriptions.	Use a predefined dictionary to map emojis to text descriptions. Example: emoji_dict = {":)": "smiley" followed by replacing emojis in the text.
	Split Segmentation	Handling missing spaces in social media text to ensure proper tokenization.	Use custom functions to detect and correct missing spaces in text. Example: Detect common patterns where spaces might be missing and insert spaces accordingly.
	Case Folding	Converting all text to the same case (usually lowercase) to ensure uniformity.	Use the .lower() method in Python to convert text to lowercase. Example: text.lower().
	Tokenization	Splitting text into individual tokens (words) to simplify text processing.	Use the NLTK word_tokenize function to split text into tokens. Example: tokens = word_tokenize(text).

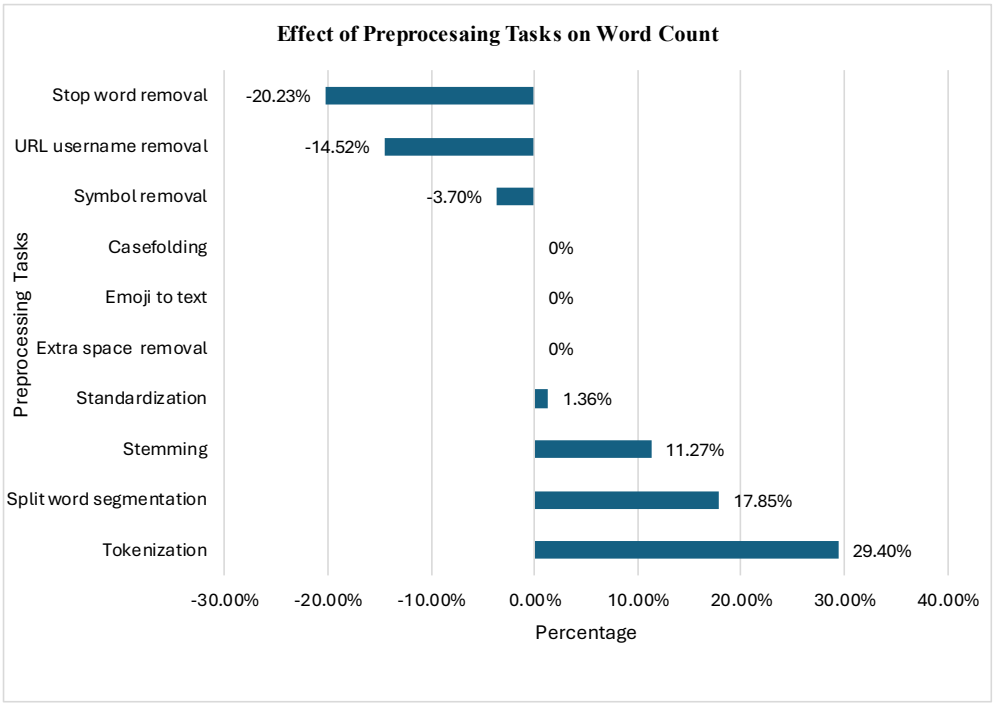

Fig. 3: Effect of Preprocessing Tasks on the Word Count

Fig. 3 illustrates the effect of the preprocessing task on the word count. Interestingly, the word count was reduced or expanded depending on the nature of the preprocessing task. If the task involves removing content, it reduces the word count. Conversely, the word count expands if the task involves replacing and splitting words. However, tasks involving simple regular expressions, such as extra space removal, emoji-to-text conversion, and case folding, result in an unchanged word count.

3.3 Feature Extraction

Feature extraction is important in text classification, which impacts the classification accuracy significantly [32]. The objectives are to capture the data pattern and reduce the data dimensionality, which helps to improve the model's ability to generalize unseen data. The process involves extracting data characteristics from the dataset [33]. The extraction process involved transforming raw data into numerical features that machine learning algorithms can interpret [34]. This transformation, also known as text vectorization, is essential for effective text classification. Researchers widely use the TF-IDF (Term Frequency-Inverse Document Frequency) feature extraction technique in text classification, as it weighs the terms in a document based on three key components: the Term Frequency (TF) factor, the Inverse Document Frequency (IDF) factor, and document length normalization [35]. The primary objectives of this technique are to assign greater significance to terms specific to a particular class (TF) and to reduce the importance of terms that occur frequently across the entire corpus (IDF) [8].

For this study, TF-IDF is employed as an extraction technique that is used as input for the classification model. The process is carried out as follows:

(a) Preprocessing Text Data: The text data was cleaned and tokenized to remove noise such as special characters, stopwords, and punctuation, ensuring meaningful features were extracted.
(b) Calculating Term Frequency (TF): TF was computed to determine how often a term appeared in each document relative to the total number of terms in that document.
(c) Calculating Inverse Document Frequency (IDF): IDF was used to reduce the weight of terms that appear frequently across all documents, ensuring that common words had less influence on the classification.
(d) Combining TF and IDF: The TF and IDF values were multiplied to assign a weight to each term, highlighting terms that were significant to a particular class while diminishing the impact of frequent, less informative terms.

3.4 Classification

The Extra Tree (ET), also known as an extremely randomized tree classifier, is an ensemble learning method that fits multiple decision trees on various sub-samples of the dataset. The objective is to utilize an efficient classification model using the Extra Tree classifier to improve predictive accuracy and control over-fitting. This ensemble learning method is superior to single classifiers because it constructs a set of classifiers and then classifies new data points by taking their weighted votes [36]. For these reasons, the Extra Tree (ET) was employed due to its computational efficiency and robustness in handling complex data patterns. Unlike Random Forest (RF), Extra Tree trains on the entire dataset without bootstrap sampling, reducing bias and improving accuracy.

To evaluate the extra tree robustness, this study compared various classification algorithms. This study experimented with six different classification algorithms to determine the best-performing classifier. Additionally, this study conducted significant experiments to implement approaches similar to our work in terms of study aim, preprocessing, and techniques. This study uses them as baseline classifiers for the performance comparison. This study compared the first approach with and without preprocessing and identified the work most similar to ours.

3.5 Study Baseline

This study performed baseline tests using the Python machine-learning library. The baseline test compared the proposed approach's performance with that of the most similar existing technique as a baseline classifier.
Table 4 includes three existing approaches: Desrul and Romadhony [4], Hendrawan, et al. [5], Isa, et al. [9], which this study carefully implemented for baseline comparison due to their similarity to this study's techniques and focus on offensive text classification. Desrul and Romadhony [4], Hendrawan, et al. [5] focused on offensive text classification in Indonesian because of the lack of available works for Malay offensive text and the linguistic similarities between Indonesian and Malay. Isa, et al. [9] also specifically addressed Malay offensive text, similar to this study.

Table 4: Summary of selected existing studies as baselines

	Authors	**Isa_SVM**	**Desrul_SVM**	**Hendrawan_RFCC**	***This Study***
	Features	[9]	[4]	[5]	
Preprocessing Task	Stop words removal	●	●		
	Stemming	●	●		
	Tokenization		●		●
	Case folding	●			●
	Punctuation and symbol removal	●	●	●	●
	Number removal	●		●	
	Emoji removal	●			
	Extra space removal				●
	Translation	●			
	URL, username removal				●
	Split hash word				●
	Emoji to text				●
	Spelling correction	●		●	●
	Replace abbreviation	●	●	●	●
	Dialect standardization				●
Feature Extraction		TF-IDF	BoW	TF-IDF	TF-IDF
Classification		SVM	SVM	RFCC	ET

Columns and acronyms: Feature Extraction: Term Frequency Inverse Document Frequency (TF-IDF), Bag of Words (BoW). Classification: Extra Tree (ET), Support Vector Machine (SVM), Random Forest Classifier Chain (RFCC)

This study selected the Extra Tree classifier for its computational efficiency and lower bias. Unlike the Random Forest, which chooses the most optimal split at each node from subsets of observations with unique features, the Extra Tree classifier utilizes the entire dataset when constructing trees. This approach helps to reduce bias, as different subsets can lead to varying biases. Therefore, the study proposed using the Extra Tree classifier to prevent this issue by sampling the entire dataset. To conduct a baseline assessment, the study employed the scikit-learn machine learning library with the default parameters for the classifier. Additionally, the parameters were fine-tuned to achieve the best results. For a fair comparison, the study used a self-collected MOFFET dataset.

3.6 Evaluation

In this section, this study defines the key performance metrics used to evaluate the classification model, including accuracy, precision, and recall. The following parameters construct the performance metrics in the context of our domain:

(a) True Positives (TP): The number of non-offensive texts correctly classified as non-offensive.
(b) True Negatives (TN): The number of offensive texts correctly classified as offensive.
(c) False Positives (FP): The number of non-offensive texts incorrectly classified as offensive
(d) False Negatives (FN): the number of offensive texts incorrectly classified as non-offensive.

Evaluating a text classification model involves three key metrics: Accuracy, precision, and recall. Below are the descriptive details of the experiment this study conducted.

(a) Accuracy(A)
Accuracy represents classification accurately identifying the offensive and non-offensive texts, also known as the consistency of the actual results for the number of samples tested.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{1}$$

(b) Precision(P)
Precision is the percentage of text identified as genuinely offensive. Precision is useful when the false positive rate is high, and it shows how much of the text is labelled as offensive or actual offensive texts.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2}$$

(c) Recall(R)
Recall, also known as sensitivity or true positive rate, measures the proportion of actual offensive text correctly identified. It assesses the model's ability to detect offensive content. If the recall is high, it means that the classifier misses fewer offensive texts.

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{\mathrm{TP.}}{\mathrm{P}} \tag{3}$$

4.0 EXPERIMENTAL RESULTS AND DISCUSSIONS

This section presents the experimental findings, analyzing the performance of different classifiers and preprocessing methods using accuracy, precision, and recall as evaluation metrics. This study discusses the impact of various preprocessing techniques such as standardization, stop word removal, and stemming. Additionally, we explore the trade-offs between data size and execution time, highlighting scenarios where metrics become more informative. This study split the experiment into 3 experiments with the following details:

(a) Experiment 1: Preprocessing for Malay Offensive Text Detection

Objective: To utilize an Extra Tree classifier for effective preprocessing in the detection of Malay offensive text.
Methodology: The dataset underwent basic preprocessing steps, which included removing URLs and usernames, eliminating extra whitespace, and performing tokenization. This study developed seven scenarios to evaluate the impact of different preprocessing techniques. These scenarios consisted of the following: standardization, no

standardization, stemming, no stemming, stopping word removal, no stopping word removal, and a combination of standardization, stemming, and stopping word removal. This experiment aimed to assess the practicality of standardization for noisy text and to explore the relationship between stemming and stopping word removal in the detection results. Based on our findings, this study selected the most suitable preprocessing setting for this research.
Expected Findings: The experiment aims to provide insights into the most effective preprocessing techniques for improving the detection of Malay offensive text.

(b) Experiment 2: Classifying Malay Offensive Text with Different Algorithms

Objective: To compare the performance of six machine learning algorithms in classifying Malay offensive text.
Methodology: In this experiment, this study compared six different machine learning algorithms to assess their effectiveness in classifying Malay offensive text. Each algorithm's performance was evaluated using accuracy, precision, and recall as the primary metrics. This comprehensive comparison provided insights into the strengths and weaknesses of each algorithm in the context of our specific application.
Findings: The results identified the most suitable algorithms for Malay offensive text classification, guiding future research and implementation efforts.

(c) Experiment 3: Baseline Studies Comparison Using the MOFFET Dataset

Objective: To compare three baseline studies using the MOFFET dataset by re-implementing a similar environment from the research paper.
Methodology: This experiment re-implemented the environments described in three baseline studies to facilitate a direct comparison using the MOFFET dataset. By closely mirroring the conditions and parameters outlined in the original studies, this study aimed to identify differences and similarities in the results. This comparison provided a benchmark for evaluating our findings against established research.
Findings: The experiment highlighted the relative performance of our approach compared to the baseline studies, offering a valuable reference for future research.
Our MOFFET dataset comprised 3,847 non-offensive texts and only 1,153 offensive texts, resulting in a significant imbalance. To address the class imbalance in the MOFFET dataset, this study employed the Synthetic Minority Oversampling Technique (SMOTE). This approach effectively balanced the dataset, ensuring unbiased classification results.

4.1 Preprocessing Selection Effect

This section provides a detailed examination of powerful preprocessing techniques in text classification. This study discusses the objectives and motivations behind this experimental study, emphasizing the need to assess the effects of preprocessing on classification performance. This study presents an experiment with various scenarios to identify the effect of preprocessing tasks especially standardization, stemming, and stop word removal on classification results. Table 5 shows the seven scenarios in our experiment, utilizing well-known preprocessing techniques in NLP. The preprocessing stage includes:

1. Basic preprocessing
2. Standardization only
3. Without stop word removal
4. Without stemming
5. Combination of standardization, stemming, and stop word removal
6. Stemming only
7. Stop word removal only
8. Without standardization

The results indicate that TF-IDF feature extraction and standardization techniques achieved an outstanding classification accuracy of 93.84%. Standardization effectively normalized noisy text, abbreviations, and Malaysian dialects into a standard form, as Dewan Bahasa dan Pustaka stated, facilitating TF-IDF's ability to quantify word relevance. The lowest accuracy, 92.11%, was observed with stemming and stop word removal without standardization. Other combinations involving stemming, stop word removal, and standardization showed very acceptable results. However, the decrease in performance was due to TF-IDF's inability to differentiate essential and less critical words after applying stemming and stop word removal.

This study found that stemming can cause word ambiguity, leading to errors such as converting '*kemaluan*' (private part) to '*malu*' (shy). Removing stop words also reduced the amount of information in the text. For example, '*kau*' (you) in the sentence *'Kau memang tidak pandai*' (You are not clever) becomes *'memang pandai*' (so clever) after stop word removal, changing the sentence's meaning. Although previous studies have deemed

stop words ineffective and meaningless for text classification tasks, removing them can alter the text's meaning and affect classification accuracy.

Our results show that the impact of stop word removal and stemming on accuracy is only noticeable when standardization is not applied. Consequently, to reduce data dimensionality, this study chose to ignore stop word removal and stemming in our preprocessing.

Table 5: Classification result

Method	Feature extraction	Scenario	Preprocessing			Acc (%)
			Standardization	Stemming	Stop words removal	
Extra Tree	TF-IDF	1	-	-	-	92.93
		2	Y	-	-	**93.84**
		3	Y	Y	-	93.37
		4	Y	-	Y	92.93
		5	Y	Y	Y	92.59
		6	-	Y	-	93.41
		7	-	-	Y	92.89
		8	-	Y	Y	92.11

Table 6 represents the effect of preprocessing on the dataset size and execution time, focusing on standardization, stemming, and stop word removal. The three preprocessing tasks were prioritized based on their significance in improving data quality. For instance, standardization for normalizing noisy text, abbreviations, and dialectal variation, stemming reduces word roots, and stopword removal removes common words and then reduces dimensionality. This study found that the MOFFET dataset contained many abbreviations and noisy words. Standardization expanded these abbreviations and noisy words to their original form, increasing the data size by approximately 2.28%. As the dataset size increases, the execution time also tends to increase. However, surprisingly, standardization required less execution time (3.24s) compared to the scenario without standardization (4.9s).

Additionally, stemming resulted in a 2.34% reduction in data size, from 1022KB to 998KB, with no change in execution time. Stop word removal significantly reduced the dataset size by about 16.50%, from 939KB to 784 KB. The execution time for stop word removal with standardization was slightly lower (4.8s) than without standardization (5.2s).

It is noteworthy that standardization, whether combined with or without stemming and stop word removal, reduced the execution time required to train the model and decreased computational complexity. In conclusion, the relationship between data size and execution time typically involves increased execution time as the dataset grows. However, this relationship can vary based on factors such as data quality and algorithmic complexity. Optimizing algorithms and employing efficient data processing techniques are essential strategies for managing and mitigating the impact of larger datasets on execution time.

Table 6: Dataset size, the data execution time of additional standardization, stemming, and stop words removal.

Preprocessing Method	With Standardization		Without Standardization	
	Data size(kb)	Execution time(s)	Data size(kb)	Execution time (s)
Basic preprocessing	**1297**	**3.24**	**939**	**4.90**
Stemming	1022	4.90	998	4.90
Stop words removal	758	4.80	784	5.20
Stemming and stop words removal	752	4.00	740	5.00

4.2 Classification Performance Comparison

In this experiment, this study demonstrated the mean and standard deviation (MSD) of our six classifiers through cross-validation of the dataset. The aim was to determine the most stable classifier among the tested classifiers. Table 7 describes the average accuracy and mean standard deviation results of six types of classifiers, utilizing K-

fold cross-validation to estimate model stability on new data. This study applied 10-fold cross-validation, as k=10 is commonly used in machine learning and offers better model performance, especially with small datasets.
By applying additional standardization preprocessing steps and the TF-IDF feature extractor, our classifier achieved the highest accuracy of 94.43% along with an exceptionally low mean standard deviation of 0.6%. The low standard deviation indicates the classifier's ability to identify the target accurately. The SVM and RF classifiers followed, achieving accuracies of 91.97% and 91.48%, respectively. Meanwhile, the KNN, LR, and BNB classifiers obtained accuracies of 87.92%, 86.50%, and 86.02%, respectively.

Table 7: K-Fold Cross Validation Accuracy and Standard Deviation result with standardization.

Method	10-Fold accuracy (%)	Standard deviation (%)
Extra Tree	**94.43**	**0.60**
Random Forest	91.48	0.90
SVM	91.97	0.90
KNN	87.92	1.40
LR	86.50	1.20
BNB	86.02	1.20

Fig. 4 illustrates the AUC-ROC curve to visualize the performance of multiple classification models. AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) is a performance measurement for classification problems at various threshold settings. AUC represents the measure of separability, and ROC is a probability curve. The ROC curve interprets the true positive rate (TPR or sensitivity) and false positive rate (FPR or specificity) near 1.

The Extra Tree classifier achieves the highest AUC value of 0.982, indicating its ability to distinguish between offensive and non-offensive text. This highest performance is due to its robustness to residual noise and variations, also with the TF-IDF vectors, particularly beneficial, leveraging the most relevant features during training contributes to classification accuracy. The second highest AUC, achieved by the SVM, is 0.971, demonstrating robust performance but slightly lower than the Extra Tree classifier. The Random Forest classifier follows with an AUC of 0.964, slightly less than SVM.

The k-nearest Neighbor and Logistic Regression classifiers achieve the lowest AUC values, 0.891 and 0.924, respectively, indicating their limited ability to distinguish between offensive and non-offensive instances. In conclusion, the Extra Tree classifier is the most effective for classifying offensive and non-offensive classes, making it suitable for the MOFFET dataset compared to other classifiers.

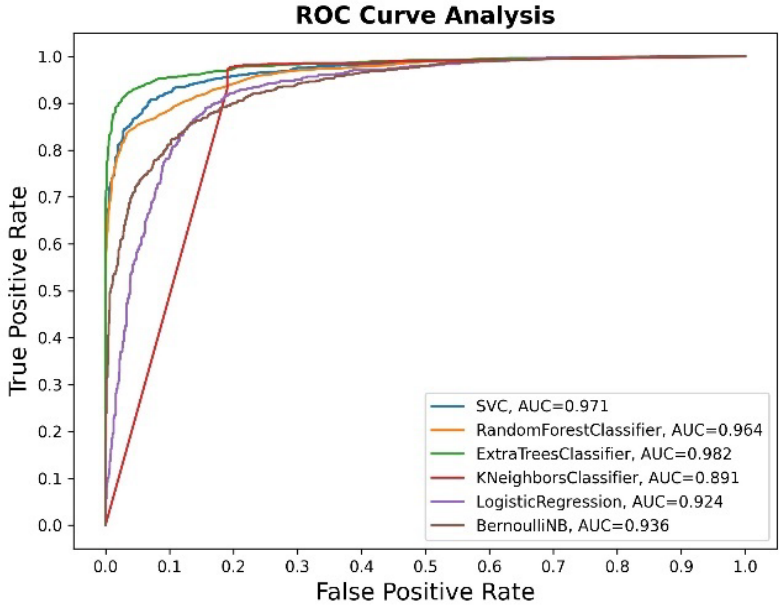

Fig. 4: AUC-ROC Curve Analysis

4.3 Comparative Study

In this experiment, this study presented the empirical results of our proposed classifier. This study compared our classifier with previous methods, such as Desrul and Romadhony [4] and Hendrawan, et al. [5] on the MOFFET dataset to achieve these empirical results. This study measured three evaluation metrics, outlined previously in the Methodology section, to ensure these methods performed better. The MOTEC framework's key advantage lies in its ability to leverage preprocessing techniques and robust classification.

Table 8 shows the method comparison between Isa_SVM, Desrul_SVM, Hendrawan_RFCC, and our MOTEC framework. This table compares the methods based on accuracy, precision, and recall. By examining the results obtained from each method, our model's accuracy is exceptionally high compared to the Desrul_SVM, Hendrawan_RFCC, and Isa_SVM methods. The consistent accuracy, precision, and recall indicate that our model's overall performance is more reliable and accurate. This advantage demonstrates the MOTEC framework's ability to provide stable and accurate classification results, which are crucial for practical applications. Desrul_SVM showed competitive results, with 11.51% lower accuracy and recall than our model, but slightly higher precision.

Meanwhile, Hendrawan_RFCC exhibited a potential trade-off between higher precision and recall but lower accuracy. This result indicates that random forest and classifier chains are unable to predict offensive and non-offensive labels proportionately due to weak label dependencies and insufficient training data. This study framework overcomes this limitation through a balanced approach that can handle class imbalances. Lastly, Isa_SVM had the lowest accuracy, precision, and recall at 70.95%, 71.90%, and 70.85%, respectively. The model's use of a random undersampling technique caused a loss of information, which our framework avoids through its comprehensive preprocessing and balanced data representation.

Table 8: Current method comparison

Model	**Method**	**Acc (%)**	**Precision (%)**	**Recall (%)**
Isa_SVM [9]	SVM	70.95	71.90	70.95
Desrul_SVM [4]	SVM	82.64	85.84	82.64
Hendrawan_RFCC [5]	RFCC	73.33	100	92.66
This study	ET	**94.15**	94.15	94.15

Table 9 compares the MOTEC framework with existing studies regarding preprocessing, execution time, and error rate results. A detailed comparison of execution times and error rates demonstrated the MOTEC framework's efficiency in real-world applications. This study approach demonstrated the lowest execution time (2.54s) and error rate (0.05%) compared to Desrul_SVM, Hendrawan_RFCC, and Isa_SVM after implementing our Malay standardization. This efficiency highlights the MOTEC framework's ability to process texts quickly without compromising accuracy, achieving a desirable balance between computational efficiency and accurate classification.

Although the Desrul_SVM approach minimized the use of preprocessing tasks, resulting in a low execution time, it had a higher error rate of 0.12% compared to this study. This outcome suggests that while the model optimized its architecture, it had a high misclassification rate. The MOTEC framework's advantage lies in its ability to maintain low error rates without sacrificing computational efficiency. The methods from Hendrawan_RFCC and Isa_SVM exhibited the highest execution times and error rates, exceeding 3 seconds and 0.20%, respectively. These results indicate that these models were unsuccessful in proportionately classifying offensive and non-offensive instances, making them unsuitable for real-time classification. In contrast, the MOTEC framework strikes a desirable balance between speed and accuracy, a critical advantage for deploying offensive text detection systems in practical settings.

Table 9: Features comparison of execution time and error rate with the existing model

Model	**Isa_SVM**	**Desrul_SVM**	**Hendrawan_RFCC**	***This Study***
Error Rate (%)	0.29	0.17	0.26	**0.05**
Execution time (s)	3.16	2.90	6.46	**2.54**

5 CONCLUSION

This study investigated existing methods for classifying Malay offensive text to promote a harmonious online environment. This study proposed Malay standardization for the additional preprocessing task instead of normalization and combined TF-IDF and an extra tree classifier to identify Malay offensive text.

This study offers several contributions. First, this study built the MOFFET dataset, containing 1,153 offensive and 3,847 non-offensive texts. Second, we created a Malay language standardization dictionary that normalizes words into their standard forms, including noisy language, Malaysian dialects, and Malay abbreviations. During the preprocessing data phase, this study applied various basic preprocessing steps to the MOFFET datasets, such as cleaning, standardization, and transformation. In a second experiment, this study evaluated the effect of preprocessing tasks, particularly standardization, stemming, and stop word removal. Our experimental results show that standard preprocessing with standardization significantly improves accuracy, while stemming and stop word removal slightly reduce classification accuracy from 93.84% to 93.41% and 92.46%, respectively. Additionally, our Malay standardization accommodates informal and misspelled words. It adapts easily to common Malay writing styles, making it a useful tool for Malay offensive language standardization and promoting a harmonious online environment.

Third, this study compared several well-known classifiers to determine the most stable classifier for the MOFFET dataset. The extra tree classifier achieved the highest accuracy, 94.43%, with an incredibly low mean standard deviation (0.6%) when combined with additional standardization preprocessing steps and the TF-IDF feature extractor. Lastly, this study conducted a comparative study using two previous approaches using the MOFFET dataset. The MOTEC model achieved an excellent accuracy of 94.15%, followed by Desrul_SVM at 82.64%, Hendrawan_RFCC at 73.33%, and Isa_SVM at 70.95%.

For future work, this study plans to explore other text classification techniques, such as BERT for the Malay language, in combination with developing an automatic update for the Malay standardization dictionary and enlarging the MOFFET corpus, as proposed by Eke, et al. [37] and Miao, et al. [25]. The MOTEC framework could also be extended to languages like Brunei or Indonesia by tailoring dictionaries and preprocessing steps to reflect their specific linguistic characteristics, such as informal word usage and code-mixing. For example, the framework could involve creating dictionaries to address Malay-English code-mixed texts and region-specific slang, ensuring accurate normalization and classification. These adaptations would enable the MOTEC framework to address linguistic nuances effectively, making it a valuable tool for offensive text classification in under-resourced languages globally. Further implementation of this framework could contribute to reducing online toxicity, making environments more welcoming to social media users.

REFERENCES

[1] S. A. Abdullah. "Safeguard Children from Cyberbullying." https://www.cybersecurity.my/data/content_files/11/1239.pdf?.diff=1393218174 (accessed 1 November 2024).

[2] D. A. Razak. "Cyberbullying Can Be Fatal Too." New Straits Times. https://www.nst.com.my/opinion/columnists/2020/03/571459/cyberbullying-can-be-fatal-too (accessed 1 November 2024).

[3] S. S. Salim, A. N. Ghanshyam, D. M. Ashok, D. B. Mazahir, and B. S. Thakare, "Deep LSTM-RNN with Word Embedding for Sarcasm Detection on Twitter," in *2020 International Conference for Emerging Technology (INCET)*, 5-7 June 2020, pp. 1-4, https://doi.org/10.1109/INCET49848.2020.9154162

[4] D. R. K. Desrul and A. Romadhony, "Abusive Language Detection on Indonesian Online News Comments," in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 5-6 December 2019, pp. 320-325, https://doi.org/10.1109/ISRITI48646.2019.9034620

[5] R. Hendrawan, Adiwijaya, and S. A. Faraby, "Multilabel Classification of Hate Speech and Abusive Words on Indonesian Twitter Social Media," in *2020 International Conference on Data Science and Its Applications (ICoDSA)*, 5-6 August 2020, pp. 1-7, https://doi.org/10.1109/ICoDSA50139.2020.9212962

[6] N. F. B. Johari and J. Jaafar, "A Malay Language Cyberbullying Detection Model on Twitter using Supervised Machine Learning," in *2022 International Visualization, Informatics and Technology Conference (IVIT)*, 1-2 November 2022, pp. 325-332, https://doi.org/10.1109/IVIT55443.2022.10033395

[7] E. Haddi, X. Liu, and Y. Shi, "The Role of Text Pre-processing in Sentiment Analysis," *Procedia Computer Science,* vol. 17, 1 January 2013, pp. 26-32, https://doi.org/10.1016/j.procs.2013.05.005

[8] O. Ying, M. Zabidi, N. Ramli, and U. Sheikh, "Sentiment analysis of informal Malay tweets with deep learning," *IAES International Journal of Artificial Intelligence (IJ-AI),* vol. 9, 1 June 2020, pp. 212-220, https://doi.org/10.11591/ijai.v9.i2.pp212-220

[9] A. M. Isa, S. Ahmad, and N. M. Diah, "Detecting Offensive Malay Language Comments on YouTube using Support Vector Machine (SVM) and Naive Bayes (NB) Model," *Journal of Positive School Psychology,* vol. 6, no. 3, 2022, pp. 8548–8560. [Online]. Available: https://www.journalppw.com/index.php/jpsp/article/view/5121

[10] Z. Mansur, N. Omar, S. Tiun, and E. M. Alshari, "A normalization model for repeated letters in social media hate speech text based on rules and spelling correction," (in eng), *PLoS One,* vol. 19, no. 3, 2024, p. e0299652, https://doi.org/10.1371/journal.pone.0299652

[11] S. Saumya, A. Kumar, and J. P. Singh, "Filtering offensive language from multilingual social media contents: A deep learning approach," *Engineering Applications of Artificial Intelligence,* vol. 133, 1 July 2024, p. 108159, https://doi.org/10.1016/j.engappai.2024.108159

[12] A. R. Pillai and B. Arun, "A feature fusion and detection approach using deep learning for sentimental analysis and offensive text detection from code-mix Malayalam language," *Biomedical Signal Processing and Control,* vol. 89, 1 March 2024, p. 105763, https://doi.org/10.1016/j.bspc.2023.105763

[13] K. L. Gwet, "Large-Sample Variance of Fleiss Generalized Kappa," *Educational and Psychological Measurement,* vol. 81, no. 4, 2021, pp. 781-790, https://doi.org/10.1177/0013164420973080

[14] C. L. Mitchell, *Language as an instrument of national policy: The Dewan Bahasa dan Pustaka of Malaysia*. The University of Wisconsin-Madison, 1993.

[15] M. A. H. b. Sazali and N. B. Idris, "Neural Machine Translation for Malay Text Normalization using Synthetic Dataset," in *2022 10th International Conference on Information and Communication Technology (ICoICT)*, 2-3 August 2022, pp. 386-390, https://doi.org/10.1109/ICoICT55009.2022.9914841

[16] S. Omar, J. A. Bakar, M. M. Nadzir, N. H. Harun, and N. Yusoff, "Text simplification for Malay corpus: A Review," in *2021 International Conference on Computer & Information Sciences (ICCOINS)*, 13-15 July 2021, pp. 345-350, https://doi.org/10.1109/ICCOINS49721.2021.9497167

[17] K. Maity, S. Bhattacharya, S. Saha, and M. Seera, "A Deep Learning Framework for the Detection of Malay Hate Speech," *IEEE Access,* vol. 11, 2023, pp. 79542-79552, https://doi.org/10.1109/ACCESS.2023.3298808

[18] H.-S. Lee, H.-R. Lee, J.-U. Park, and Y.-S. Han, "An abusive text detection system based on enhanced abusive and non-abusive word lists," *Decision Support Systems,* vol. 113, 1 September 2018, pp. 22-31, https://doi.org/10.1016/j.dss.2018.06.009

[19] K. Nugroho *et al.*, "Improving Random Forest Method to Detect Hatespeech and Offensive Word," *2019 International Conference on Information and Communications Technology (ICOIACT),* 24-25 July 2019, pp. 514-518, https://doi.org/10.1109/ICOIACT46704.2019.8938451

[20] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, and M. T. Sadiq, "Automatic Detection of Offensive Language for Urdu and Roman Urdu," *IEEE Access,* vol. 8, 2020, pp. 91213-91226, https://doi.org/10.1109/ACCESS.2020.2994950

[21] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection," *IEEE Access,* vol. 6, 2018, pp. 13825-13835, https://doi.org/10.1109/ACCESS.2018.2806394

[22] N. Nugrahaningsih, A. Lestari, and D. Karolita, "Identifying the Offensive Words in the Twitter Using Latent Semantic Analysis," in *2020 6th International Conference on Computing Engineering and Design (ICCED)*, 15-16 October 2020, pp. 1-4, https://doi.org/10.1109/ICCED51276.2020.9415773

[23] N. A. A. Aziz, M. A. Maarof, and A. Zainal, "Hate Speech and Offensive Language Detection: A New Feature Set with Filter-Embedded Combining Feature Selection," in *2021 3rd International Cyber Resilience Conference (CRC)*, 29-31 January 2021, pp. 1-6, https://doi.org/10.1109/CRC50527.2021.9392486

[24] F. Husain and Ö. Uzuner, "Transfer Learning Approach for Arabic Offensive Language Detection System - BERT-Based Model," in *2021 4th International Conference on Computer Applications & Information Security (ICCAIS) - Contemporary Computer Technologies and Applications*, 1 February 2021, https://doi.org/10.48550/arXiv.2102.05708

[25] Z. Miao *et al.*, "Detecting Offensive Language Based on Graph Attention Networks and Fusion Features," *IEEE Transactions on Computational Social Systems,* vol. 11, no. 1, 2024, pp. 1493-1505, https://doi.org/10.1109/TCSS.2023.3250502

[26] A. Al-Saffar, S. Awang, H. Tao, N. Omar, W. Al-Saiagh, and M. Al-bared, "Malay sentiment analysis based on combined classification approaches and Senti-lexicon algorithm," *PLOS ONE,* vol. 13, no. 4, 2018, p. e0194852, https://doi.org/10.1371/journal.pone.0194852

[27] N. Zabha, Z. Ayop, S. Anawar, H. Erman, and Z. Zainal, "Developing Cross-lingual Sentiment Analysis of Malay Twitter Data Using Lexicon-based Approach," *International Journal of Advanced Computer Science and Applications,* vol. 10, 1 January 2019, https://doi.org/10.14569/IJACSA.2019.0100146

[28] N. H. Mahadzir, N. H. A. Razak, and M. F. M. Omar, "A New Sentiment Analysis Model for Mixed Language using Contextual Lexicon," in *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, 1-3 December 2020, pp. 1-5, https://doi.org/10.1109/ICRAIE51050.2020.9358286

[29] M. Sulaiman, N. Aziz, A. Zabidi, Z. Jantan, I. Yassin, and M. S. A. Megat Ali, "A Systematic Approach for Malay Language Dialect Identification by Using CNN," *Journal of Electrical & Electronic Systems Research,* vol. 19, 10 January 2021, pp. 25-37, https://doi.org/10.24191/jeesr.v19i1.004

[30] M. H. A. Hijazi, L. Libin, R. Alfred, and F. Coenen, "Bias aware lexicon-based Sentiment Analysis of Malay dialect on social media data: A study on the Sabah Language," in *2016 2nd International Conference on Science in Information Technology (ICSITech)*, 26-27 October 2016, pp. 356-361, https://doi.org/10.1109/ICSITech.2016.7852662

[31] M. Susanty, A. F. Rahman, M. D. Normansyah, and A. Irawan, "Offensive language detection using artificial neural network," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019: IEEE, pp. 350-353.

[32] R. Dzisevič and D. Šešok, "Text Classification using Different Feature Extraction Approaches," in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 25-25 April 2019, pp. 1-4, https://doi.org/10.1109/eStream.2019.8732167

[33] F. P. Shah and V. Patel, "A review on feature selection and feature extraction for text classification," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 23-25 March 2016, pp. 2264-2268, https://doi.org/10.1109/WiSPNET.2016.7566545

[34] M. Zareapoor and S. K. R, "Feature Extraction or Feature Selection for Text Classification: A Case Study on Phishing Email Detection," *International Journal of Information Engineering and Electronic Business,* vol. 7, no. 2, 8 March 2015, pp. 60-65, https://doi.org/10.5815/ijieeb.2015.02.08

[35] S. Vijayarani, M. J. Ilamathi, and M. N. S. Nithya, "Preprocessing Techniques for Text Mining-An Overview," in *International Journal of Computer Science & Communication Networks*, 2015, vol. 5, pp. 7-16.

[36] A. Khatua, A. Khatua, and E. Cambria, "A tale of two epidemics: Contextual Word2Vec for classifying twitter streams during outbreaks," *Information Processing & Management,* vol. 56, no. 1, 1 January 2019, pp. 247-257, https://doi.org/10.1016/j.ipm.2018.10.010

[37] C. I. Eke, A. A. Norman, and L. Shuib, "Context-Based Feature Technique for Sarcasm Identification in Benchmark Datasets Using Deep Learning and BERT Model," *IEEE Access,* vol. 9, 2021, pp. 48501-48518, https://doi.org/10.1109/ACCESS.2021.3068323