Enhanced Hate Speech Detection in Indonesian-English Code-Mixed Texts Using XLM-RoBERTa

Farrel Dinarta, Arya Wicaksana*
Department of Informatics, Universitas Multimedia Nusantara, Tangerang 15810, Banten, Indonesia
E-mail: farrel.dinarta@student.umn.ac.id, arya.wicaksana@umn.ac.id
*Corresponding author

Keywords: hate speech detection, Indonesian-English, XLM-RoBERTa

Received: December 1, 2024

The prevalence of hate speech on digital platforms presents significant challenges, particularly in multilingual communities where code-mixing complicates detection. This study explores the use of XLM-RoBERTa, a transformer-based model with robust multilingual capabilities, to detect hate speech within code-mixed texts, focusing on Indonesian-English code-mixing. Traditional hate speech detection models rely on single-language datasets, limiting their effectiveness in such environments. We employ a dataset consisting of Indonesian, English, and code-mixed Indonesian-English texts to evaluate XLM-RoBERTa's performance. The dataset comprises 24,844 training samples, 2,760 test samples, and an additional 100 supplementary samples. Key hyperparameters included a batch size of 16 and 32, with a learning rate ranging from 1e-5 to 5e-5. The model achieved near-perfect accuracy (99.6%) on the primary test set and demonstrated strong generalization across realistic supplementary data, achieving an F1-score of 90.94%. These findings underscore the model's potential for application in complex linguistic contexts, contributing to the development of effective code-mixed hate speech detection.

Povzetek: Raziskana je uporaba modela XLM-RoBERTa za zaznavanje sovražnega govora v besedilih z mešanjem indonezijskega in angleškega jezika, kar izboljša kvaliteto zaznavanja v večjezičnih okoljih.

1 Introduction

The rapid growth of social media and online communication platforms has significantly transformed global interactions, allowing for exchanging ideas across diverse linguistic and cultural boundaries. However, this interconnectedness has also contributed to the proliferation of harmful content, such as hate speech, which presents considerable social and ethical challenges. Detecting and moderating hate speech is essential for fostering a safe and inclusive digital environment [1], [2].

The task becomes increasingly complex in code-mixed texts, where users frequently switch between languages within a single conversation or incorporate terms from multiple languages [2]. Code-mixing is prevalent in multilingual communities and among bilingual individuals. Traditional hate speech detection systems, which primarily rely on single-language datasets and algorithms, struggle to identify harmful content effectively in these contexts. Most conventional approaches employ supervised machine learning models or rule-based systems that require extensive language-specific resources, such as labeled datasets and lexicons [3].

Code-mixed datasets are particularly challenging to obtain, as many publicly available datasets focus on individual languages, given that most people typically communicate in one language at a time. Code-mixing is more common in specific domains or platforms, such as social media and multilingual communities. While these platforms are invaluable for collecting code-mixed data, the informal and inconsistent nature of the content—characterized by acronyms, emojis, and spelling variations—adds complexity to dataset curation [4]. Moreover, linguistic diversity within such datasets is often limited to specific language pairs, further restricting their practical utility.

Despite the growing interest in hate speech detection, research focusing on code-mixed datasets remains limited. Transformer-based methods, particularly BERT-based models, have demonstrated superior accuracy in hate speech detection [5]. This study aims to address this gap by exploring the application of the XLM-RoBERTa model for detecting hate speech in code-mixed contexts, specifically in Indonesian and English. By leveraging XLM-RoBERTa's multilingual capabilities, this approach seeks to enhance the detection of harmful content in environments where language boundaries are increasingly blurred. The model's effectiveness will be evaluated using accuracy and F1-score metrics, contributing valuable insights to the development of robust hate speech detection systems in multilingual settings.

The remainder of this paper is organized as follows: Chapter 2 surveys related works; Chapter 3 presents the preliminaries; Chapter 4 outlines the methods; Chapter 5 discusses the results, contributions, and limitations; and Chapter 6 concludes the study with suggestions for future research.

2 Related works

XLM-RoBERTa has been applied to detect hate speech in code-mixed datasets. Tita et al. [6] compared the performance of mBERT and XLM-RoBERTa in detecting hate speech in English and French, including code-mixed contexts. The evaluation results show that XLM-RoBERTa achieves a higher macro-average score of 0.51, outperforming mBERT, which obtained 0.41, in English-French code-mixed scenarios.

Wang et al. [7] tested XLM-RoBERTa for offensive language detection in English, Turkish, Arabic, Danish, and Greek. The model achieved the average F1-scores of 0.9255, 0.8224, 0.9015, 0.8136, and 0.8392 for each language, respectively.

Suhartono et al. [8] performed a comparison study of how mBERT and XLM-RoBERTa works on classifying fake Indonesian news. The proposed model is proven to be successful with results of accuracy, precision, recall, and F1 of 0.9051, 0.9515, 0.8233, and 0.8828 respectively for the mBERT model with 10 topic words and 0.8935, 0.8818, 0.8712, and 0.8765 for the XLM-R model with 10 topic words. Table 1 presents the performance of XLM-RoBERTa in each language based on findings from the previous related works mentioned.

Table 1: XLM-RoBERTa performance across languages.

Works	Model	Dataset	Result
(2022) Suhartono et al. [8]	XLM-RoBERTa	Indonesian	The model achieves accuracy of 89.35%, precision of 88.18%, recall of 87.12%, and f1-score of 87.65%
	mBERT		The model achieves accuracy of 90.51%, precision of 95.15%, recall of 82.33%, and f1-score of 88.28%
(2021) Tita et al. [6]	XLM-RoBERTa	English-French	The model achieves macro average of 51%
		English	The model achieves macro average of 44%
		French	The model achieves macro average of 32%
	mBERT	English-French	The model achieves macro average of 41%
		English	The model achieves macro average of 71%
		French	The model achieves macro average of 66%
(2020) Wang et al. [7]	XLM-RoBERTa	English	The model achieves f1-score of 92.55%
		Turkish	The model achieves f1-score of 82.24%
		Arabic	The model achieves f1-score of 90.15%
		Danish	The model achieves f1-score of 81.36%
		Greek	The model achieves f1-score of 83.92%

Existing studies on XLM-RoBERTa and mBERT largely focus on other languages, often overlooking code-mixed contexts. Even when code-mixed scenarios are considered, the reported metric scores remain relatively low, indicating room for improvement. Notably, XLM-RoBERTa has demonstrated a strong ability to capture contextual nuances, including in code-mixed settings, particularly for high-resource languages like English. Bahasa Indonesia, also considered a high-resource language for XLM-RoBERTa, has approximately 22.704 million tokens compared to English’s 55.608 million [9], making it significantly better represented than most languages. This reinforces XLM-RoBERTa’s potential effectiveness in addressing code-mixed contexts, such as hate speech detection.

Since code-mixed resources are often limited or impractical to use, Large Language Models (LLMs) can also be leveraged to generate datasets for hate speech identification. Terblanche et al. [4] demonstrated the use of GPT-3.5 to generate code-mixed sentences in Afrikaans-English and Yoruba-English. Their findings showed that the generated data—written using the English alphabet and Latin script—was of high quality, with only minor grammatical errors that did not significantly affect meaning. This success suggests an opportunity to refine prompting guidelines to improve data quality, further supported by the fact that XLM-RoBERTa has been trained on a substantial amount of data in both Indonesian and English.

3 Preliminaries

XLM-RoBERTa (Cross-lingual Robustly Optimized Bidirectional Encoder Representations from Transformers Approach) is a transformer-based language model which enhances the state-of-the-art on mixedmultilingual understanding tasks through the joint pretraining large transformer models across diverse languages. This model is built upon the advancements of the RoBERTa model, an optimized version of BERT with dynamic masking, removal of *next sentence prediction* (NSP), larger mini-batches, and byte-level Byte Pair Encoding (BPE) tokenizer which relies on subword units and makes it possible to learn a subword vocabulary that can still encode any input text without introducing any unrecognizable tokens, ensuring success interpretation on new or unseen terms. The model training leveraged SPM-preprocessed text data from CommonCrawl scaled to cover 100 languages to handle diverse linguistic structures [9], [10].

The model is built on top of transformer and MLM (Masked Language Model) architecture, which excels at processing sequential data such as text by utilizing bidirectional self-attention layers [11] which helps on capturing the contextual relationship between words regardless of the language. Bidirectional pre-training mechanism allows the model to achieve state-of-the-art performance which reduces the need for various heavily-tuned task-specific architecture, and also predict or learn bidirectional context by predicting missing words to better understand hidden or implied subtle relationships and context. XLM-RoBERTa variant specifically applies *subword* tokenization directly on raw text data and utilizing sample batches from diverse languages using the same sampling distribution. This model additionally does not implement language embeddings which results in improved performance when dealing with code-mixed contexts, enabling it to learn complex patterns and structures in multiple languages, especially for mixed-code hate speech detection [12].

Figure 1 illustrates transformer encoder architecture while Figure 2 represents the multilingual MLM architecture. Both figures provide an overview of XLM-RoBERTa architecture, which utilizes transformer encoder model [9], [11]. In this model, inputs are preprocessed through the MLM which is specifically pre-trained for transformer encoder. Afterwards, the MLM predicts the original content of input tokens based on the remaining bidirectional contexts from randomly masked portions of the input [13].

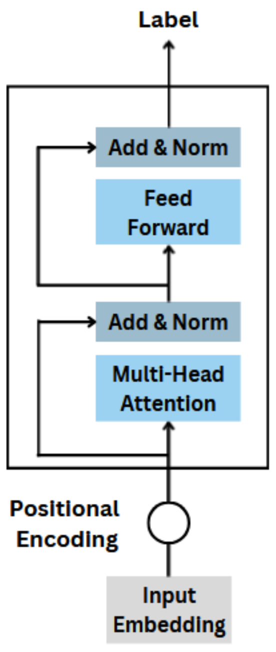


Figure 1: Transformer encoder architecture. Source: [14].

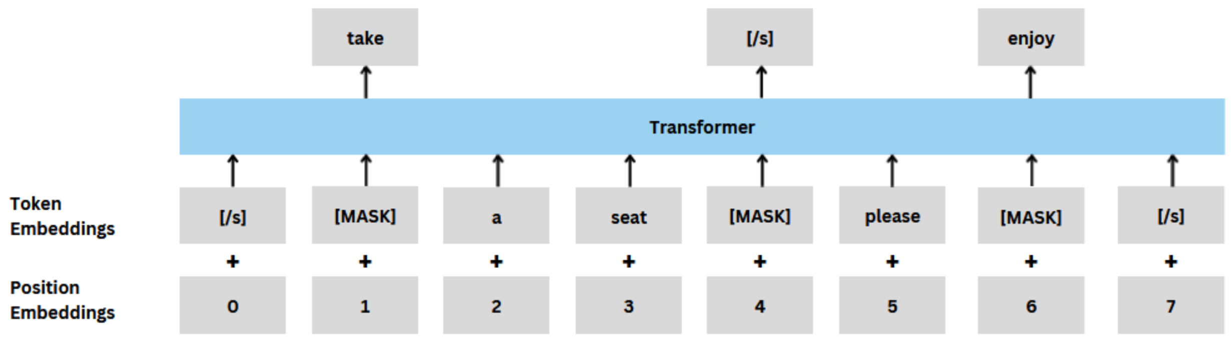
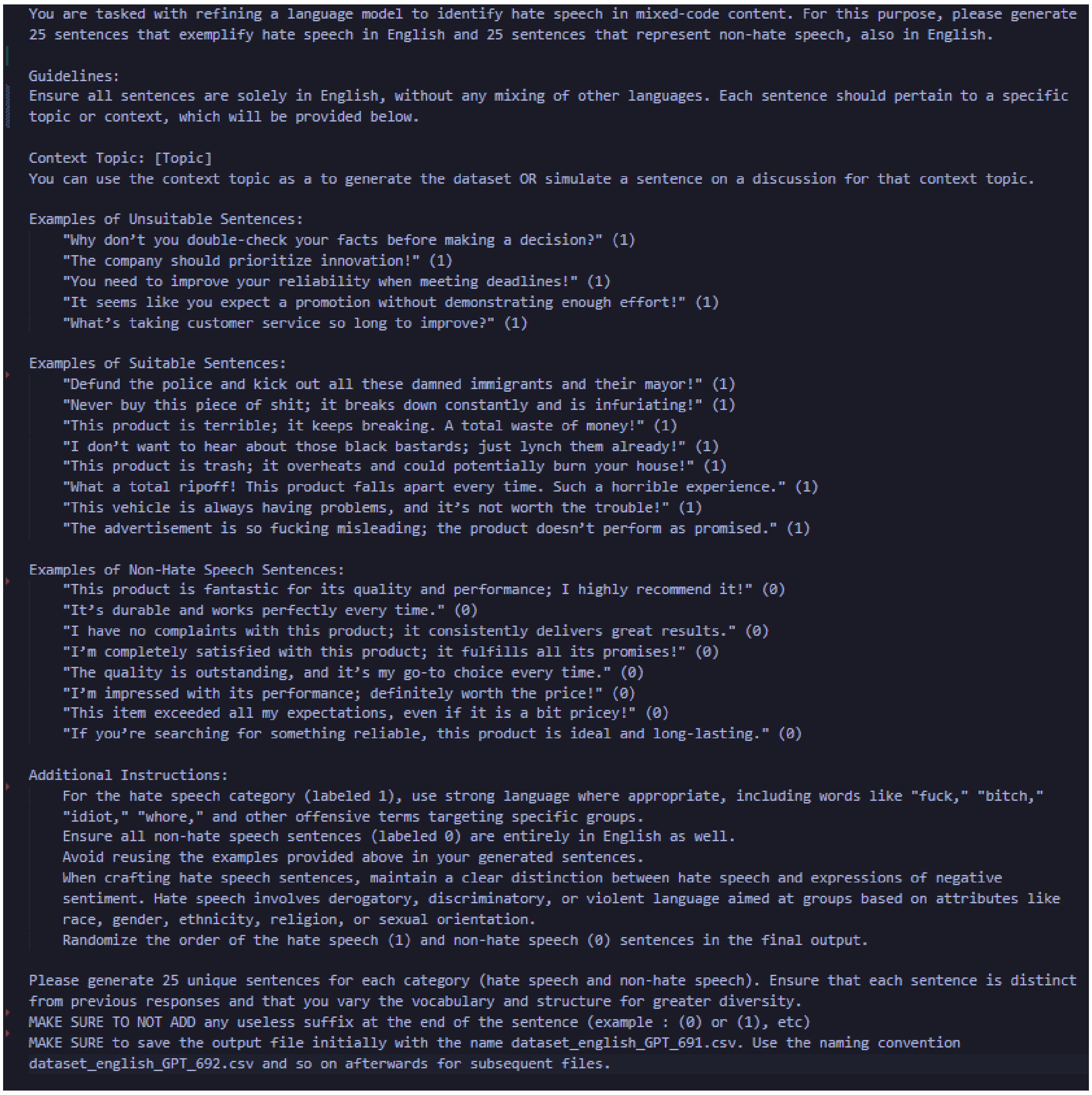


Figure 2: MLM architecture on XLM-RoBERTa. Source: [15].

4 Methods

4.1 Data collection and generation

Data collection utilizes variants of the GPT-4 large language model (LLM), including GPT-4o and GPT-4o mini, which are used interchangeably across a total of 600 prompt executions. The prompt is carefully designed with multiple detailed examples of real-life sentence scenarios to enhance contextual relevance. To ensure diversity, each execution features unique topics such as politics, religion, sports, gaming, and other common aspects of daily life. Additionally, the model's memory is periodically reset after several prompts to minimize redundancy and prevent duplicate outputs. After data generation, the dataset undergoes filtering to remove any duplicate content. Figure 3 illustrates the prompt used for data generation.



```
You are tasked with refining a language model to identify hate speech in mixed-code content. For this purpose, please generate
25 sentences that exemplify hate speech in English and 25 sentences that represent non-hate speech, also in English.

Guidelines:
Ensure all sentences are solely in English, without any mixing of other languages. Each sentence should pertain to a specific
topic or context, which will be provided below.

Context Topic: [Topic]
You can use the context topic as a to generate the dataset OR simulate a sentence on a discussion for that context topic.

Examples of Unsuitable Sentences:
    "Why don't you double-check your facts before making a decision?" (1)
    "The company should prioritize innovation!" (1)
    "You need to improve your reliability when meeting deadlines!" (1)
    "It seems like you expect a promotion without demonstrating enough effort!" (1)
    "What's taking customer service so long to improve?" (1)

Examples of Suitable Sentences:
    "Defund the police and kick out all these damned immigrants and their mayor!" (1)
    "Never buy this piece of shit; it breaks down constantly and is infuriating!" (1)
    "This product is terrible; it keeps breaking. A total waste of money!" (1)
    "I don't want to hear about those black bastards; just lynch them already!" (1)
    "This product is trash; it overheats and could potentially burn your house!" (1)
    "What a total ripoff! This product falls apart every time. Such a horrible experience." (1)
    "This vehicle is always having problems, and it's not worth the trouble!" (1)
    "The advertisement is so fucking misleading; the product doesn't perform as promised." (1)

Examples of Non-Hate Speech Sentences:
    "This product is fantastic for its quality and performance; I highly recommend it!" (0)
    "It's durable and works perfectly every time." (0)
    "I have no complaints with this product; it consistently delivers great results." (0)
    "I'm completely satisfied with this product; it fulfills all its promises!" (0)
    "The quality is outstanding, and it's my go-to choice every time." (0)
    "I'm impressed with its performance; definitely worth the price!" (0)
    "This item exceeded all my expectations, even if it is a bit pricey!" (0)
    "If you're searching for something reliable, this product is ideal and long-lasting." (0)

Additional Instructions:
    For the hate speech category (labeled 1), use strong language where appropriate, including words like "fuck," "bitch,"
    "idiot," "whore," and other offensive terms targeting specific groups.
    Ensure all non-hate speech sentences (labeled 0) are entirely in English as well.
    Avoid reusing the examples provided above in your generated sentences.
    When crafting hate speech sentences, maintain a clear distinction between hate speech and expressions of negative
    sentiment. Hate speech involves derogatory, discriminatory, or violent language aimed at groups based on attributes like
    race, gender, ethnicity, religion, or sexual orientation.
    Randomize the order of the hate speech (1) and non-hate speech (0) sentences in the final output.

Please generate 25 unique sentences for each category (hate speech and non-hate speech). Ensure that each sentence is distinct
from previous responses and that you vary the vocabulary and structure for greater diversity.
MAKE SURE TO NOT ADD any useless suffix at the end of the sentence (example : (0) or (1), etc)
MAKE SURE to save the output file initially with the name dataset_english_GPT_691.csv. Use the naming convention
dataset_english_GPT_692.csv and so on afterwards for subsequent files.
```

Figure 3: Prompt for data generation.

The Indonesian dataset consists of 9,783 entries, with 4,911 (50.2%) categorized as hate speech and 4,872 (49.8%) as non-hate speech. The English dataset includes 9,968 entries, with 4,973 (49.9%) labeled as hate speech and 4,995 (50.1%) as non-hate speech. Additionally, the code-mixed dataset contains 7,835 entries, of which 3,951 (50.4%) are designated as hate speech and 3,884 (49.6%) as non-hate speech. These figures represent the final number of data entries after the removal of duplicates to ensure data quality and consistency.

The code-mixed dataset could be retrieved from specific platforms, such as social media, which reflect real-world scenarios. Social media platforms are invaluable for gathering code-mixed data due to the extensive and diverse linguistic expressions they contain. However, the informal nature of the content, often characterized by acronyms, emojis, and spelling mistakes, presents significant challenges for effective processing [4].

Using generated data, such as datasets created with GPT, in hate speech detection introduces important ethical implications and potential biases that must be addressed. First, GPT-based models may inadvertently reproduce biases present in their training data, leading to the propagation of stereotypes or inequities in the generated dataset. This can result in a biased hate speech detection model that disproportionately misclassifies or overlooks hate speech targeting certain groups, potentially reinforcing societal prejudices. However, this bias may be intentionally leveraged to ensure the model detects specific patterns, aligning with the main objective of hate speech identification. Second, the synthetic nature of the data might lack the nuanced and context-specific complexity of real-world hate speech, which could reduce the model's effectiveness in handling real-world scenarios. Therefore, rigorous manual evaluation, curation, and refinement of the generated dataset—such as selfembedding hate keywords in certain datasets—are essential to ensure its quality, fairness, and relevance.

4.2 Data preprocessing

Figure 4 illustrates the complete sequence of preprocessing steps involved in preparing the data for model input. This process includes cleansing the raw text, tokenizing it into distinct tokens, and applying padding and truncation based on the 95th percentile of sequence lengths in the dataset. Finally, masking is applied to differentiate between empty and non-empty tokens, aiding the model in processing the text effectively.

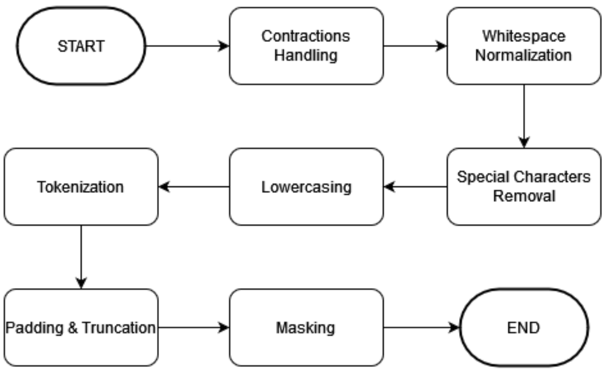


Figure 4: Data preprocessing steps.

Figure 4 outlines the data preprocessing pipeline for training XLM-RoBERTa in the context of hate speech detection for Indonesian-English code-mixed text. The process begins with contraction handling, where common abbreviations and informal contractions (e.g., "don't" → "do not", "*gak*" → "*tidak*") are expanded to their standard forms, ensuring better text representation. Next, whitespace normalization standardizes spaces and removes unnecessary gaps, followed by special character removal to eliminate symbols, emojis, or non-text elements that do not contribute to meaning.

The text is then lowercased to ensure consistency, especially for models that are case-sensitive. Tokenization follows, where the text is split into subword units using SentencePiece, the tokenization method employed by XLM-RoBERTa, enabling better handling of multilingual and code-mixed text. Subsequently, padding and truncation are applied to standardize sequence lengths, preventing excessive memory usage and maintaining uniform input dimensions. Masking is performed as part of the Masked Language Model (MLM) objective, where certain tokens are randomly replaced with a mask token, helping the model learn contextual relationships. Finally, the processed data is fed into the model for training and evaluation.

This preprocessing pipeline ensures that XLM-RoBERTa effectively learns language patterns in code-mixed Indonesian-English text, improving its ability to detect hate speech accurately while handling the linguistic variations commonly present in informal online discourse. By standardizing the text and preparing it for optimal processing, each step contributes to the model's enhanced understanding of both the individual languages involved and their intermingling in code-mixed contexts. As a result, the model is better equipped to identify subtle patterns and nuances in the text, making it more effective for real-world hate speech detection tasks.

4.3 Hyperparameter tuning

The pre-trained XLM-RoBERTa-Base model undergoes fine-tuning with variations in key hyperparameters to optimize performance by implementing grid search. Specifically, the batch size is tested with values of 16 and 32, while the learning rate is adjusted within the range of 1e-5 to 5e-5. With 10 parameter combinations (2 batch sizes × 5 learning rates), the grid search achieved efficient coverage of the hyperparameter space, providing an effective yet computationally feasible approach to optimizing the model for the task.

The selection of hyperparameter ranges for fine-tuning the XLM-RoBERTa-Base model is guided by the need to balance computational efficiency and performance optimization. Batch sizes of 16 and 32 are chosen to examine the trade-off between gradient update precision and memory consumption. Smaller batch sizes provide more precise updates but require a greater number of iterations, whereas larger batch sizes accelerate training at the potential cost of less precise convergence. The learning rate range of 1e-5 to 5e-5 is selected based on best practices for transformer-based models, ensuring stable convergence at lower rates while allowing faster training at higher rates without overshooting the optimal minima. This approach strikes a balance between preserving pre-trained weights and adapting to the hate speech detection task, where capturing subtle linguistic nuances is crucial.

An early stopping mechanism with checkpointing is implemented to prevent overfitting and enhance generalization. This mechanism monitors validation loss over five epochs and restores the best-performing weights, ensuring optimal model performance without excessive computational overhead. The number of training epochs is also tuned through an adaptive monitoring process. Training begins with a minimum of five epochs, and if validation loss does not decrease after five additional epochs beyond the current best checkpoint, the model reverts to the last optimal weights. If validation loss improves, the new epoch is designated as the best checkpoint, and monitoring restarts. This tuning strategy achieves an optimal balance between convergence speed and model stability, ensuring efficient adaptation of the XLM-RoBERTa-Base model to the task of mixed-code hate speech detection.

The chosen hyperparameter combinations were validated by calculating performance metrics, including F1-score and accuracy, for each configuration (learning rates ranging from 1e-5 to 5e-5 and batch sizes of 16 and 32) on the validation set. These metrics were evaluated across multiple training runs to account for variability introduced by random initialization and data splits.

Table 2 demonstrates that the optimal model configuration is achieved with a learning rate of 2e-5, a batch size of 16, and at epoch 8, resulting in a validation loss of 0.0355. This configuration also yields high validation accuracy of 99.23%, validation precision of 99.23%, validation recall of 99.23%, and validation F1 score of 99.23%, indicating strong model performance across multiple metrics. Minimizing validation loss is crucial as it reflects the model's capacity to generalize effectively to unseen data, thus mitigating overfitting and ensuring robust performance beyond the training dataset.

Generally, models with lower validation loss are associated with better generalization, making it a reliable criterion for model selection. Additionally, the results indicate that a lower learning rate of 1e-5, 2e-5, and 3e-5 outperforms higher rates such as 4e-5 and 5e-5, where smaller batch sizes showing a slight advantage in this context, as seen in the top-performing models. This outcome is attributed to the fact that lower learning rates enable gradual and precise convergence, minimizing the risk of overshooting optimal solution which is a common issue with higher learning rates. High learning rates can lead to unstable training dynamics, as evidenced by increased losses for both training and validation at learning rates of 4e-5 and 5e-5.

In conclusion, a low learning rate contributes to stable training, reducing both training and validation losses, thereby enhancing the model's overall performance and generalization capability. Table 3 shows training and validation progress over epochs for the chosen model.

Table 2: Hyperparameter tuning results ranked by validation loss.

Learning Rate	Batch Size	Best Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy	Train Precision	Validation Precision	Training Recall	Validation Recall	Training F1 Score	Validation F1 Score
2e-5	16	8	0.0547	0.0355	0.9926	0.9923	0.9916	0.9923	0.9925	0.9923	0.9920	0.9923
1e-5	16	6	0.0549	0.0360	0.9909	0.9928	0.9917	0.9943	0.9883	0.9918	0.9900	0.9930
1e-5	32	18	0.0395	0.0552	0.9933	0.9919	0.9938	0.9923	0.9922	0.9911	0.9910	0.9917
2e-5	32	5	0.1708	0.0830	0.9716	0.9882	0.9710	0.9878	0.9596	0.9886	0.9653	0.9882
3e-5	16	5	0.2033	0.1236	0.9588	0.9852	0.9624	0.9849	0.9390	0.9857	0.9506	0.9853
3e-5	32	9	0.2632	0.1561	0.9593	0.9556	0.9594	0.9602	0.9592	0.9502	0.9593	0.9552
4e-5	32	5	0.7210	0.6838	0.5677	0.5093	0.5467	0.5093	0.5274	0.5093	0.5369	0.5093
4e-5	16	7	0.7067	0.6870	0.5029	0.6393	0.5008	0.5335	0.4827	0.9123	0.4916	0.6733
5e-5	16	11	1.3632	0.6955	0.5007	0.5093	0.5013	0.500	0.5458	1.000	0.5226	0.6667
5e-5	32	5	4.1957	4.2744	0.5016	0.4907	0.5053	0.4907	0.2449	0.4907	0.3299	0.4907

Table 3: Performance metrics over epochs.

Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy	Train Precision	Validation Precision	Training Recall	Validation Recall	Training F1 Score	Validation F1 Score
1	0.3068	0.1222	0.9464	0.9867	0.9490	0.9870	0.8871	0.9869	0.9170	0.9869
2	0.1364	0.0619	0.9785	0.9902	0.9802	0.9921	0.9765	0.9891	0.9783	0.9906
3	0.0858	0.0704	0.9846	0.9913	0.9849	0.9926	0.9827	0.9891	0.9838	0.9908
4	0.0921	0.0972	0.9876	0.9899	0.9874	0.9889	0.9866	0.9909	0.9870	0.9899
5	0.0669	0.0673	0.9898	0.9913	0.9910	0.9934	0.9883	0.9902	0.9896	0.9918
6	0.1053	0.0575	0.9842	0.9929	0.9851	0.9929	0.9809	0.9929	0.9830	0.9929
7	0.0798	0.0444	0.9887	0.9929	0.9877	0.9933	0.9880	0.9921	0.9878	0.9927
8	0.0547	0.0356	0.9926	0.9923	0.9916	0.9923	0.9925	0.9923	0.9920	0.9923
9	0.0504	0.0862	0.9867	0.9837	0.9860	0.9835	0.9895	0.9842	0.9877	0.9838
10	0.0785	0.0590	0.9876	0.9931	0.9859	0.9923	0.9882	0.9938	0.9870	0.9930

11	0.0490	0.0541	0.9915	0.9938	0.9913	0.9926	0.9907	0.9943	0.9910	0.9934
12	0.0886	0.1376	0.9892	0.9650	0.9908	0.9665	0.9815	0.9625	0.9861	0.9645
13	0.0764	0.0798	0.9708	0.9867	0.9781	0.9874	0.9744	0.9860	0.9762	0.9867

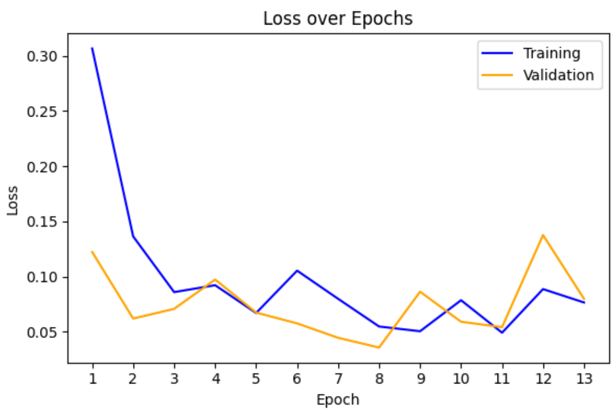


Figure 5: Training & validation loss over epochs.

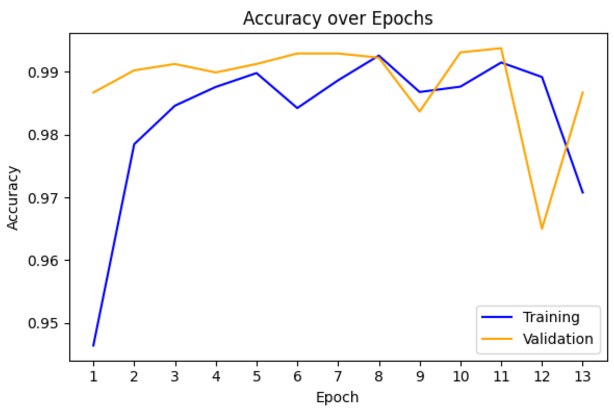


Figure 6: Training & validation accuracy over epochs.

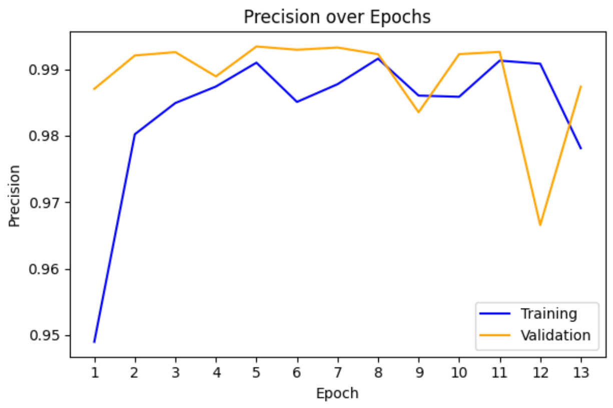


Figure 7: Training & validation precision over epochs.

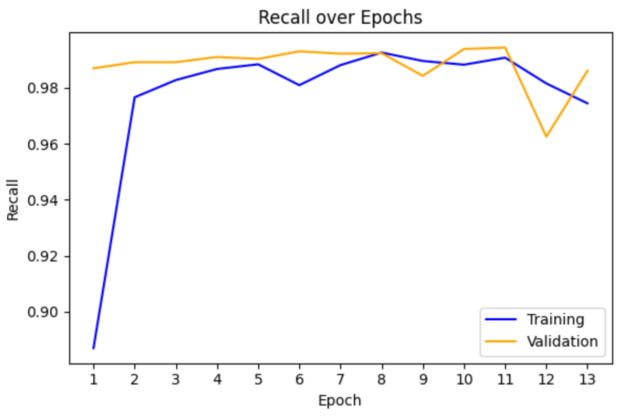


Figure 8: Training & validation recall over epochs.

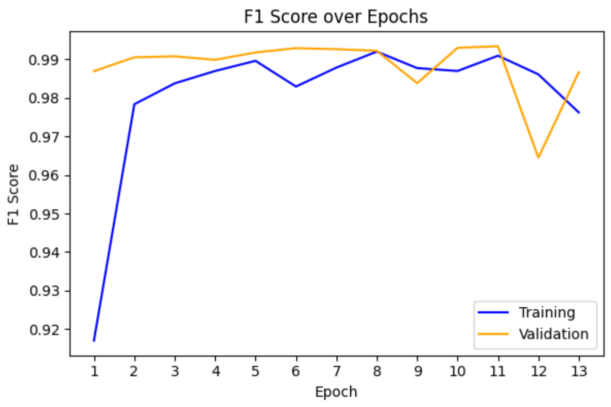


Figure 9: Training & validation f1-score over epochs.

Figures 5–9 illustrate the training and validation performance trends over 13 epochs across multiple evaluation metrics. Figure 5 shows a sharp decline in training loss during the initial epochs, followed by stabilization with minor fluctuations, while validation loss remains consistently lower, with a slight increase around epoch 12, suggesting minor variations in generalization. Figure 6 highlights the accuracy progression, where both training and validation accuracy rapidly exceed 0.98 and remain stable, indicating strong generalization.

Figure 7 presents precision trends, with training precision surpassing 0.98 early and stabilizing near 0.99, while validation precision remains consistently high with slight fluctuations. Figure 8 illustrates recall performance, where both training and validation recall remain around 0.99, with a minor dip at epoch 12, reinforcing the model's ability to minimize false negatives. Figure 9 depicts F1-score trends, showing a rapid increase followed by stabilization near 0.99 for both training and validation sets, ensuring a balanced precision-recall trade-off. Overall, these results indicate that the model effectively generalizes while maintaining high performance across all key metrics, with only minor variations observed in the later epochs.

4.4 Further evaluation

The validation of the chosen hyperparameter combinations involved calculating performance metrics, such as F1-score and accuracy, for each configuration (learning rates ranging from 1e-5 to 5e-5 and batch sizes of 16 and 32) on the validation set. These metrics were evaluated across multiple training runs to account for variability introduced by random initialization and data splits.

5 Results and analysis

Table 2 shows that lower learning rates, such as 2e-5 and 1e-5 perform better in tasks like hate speech detection involving complex, code-mixed datasets due to their ability to ensure stable convergence and precise weight updates. These rates allow the model to better capture subtle linguistic patterns and reduce the risk of overfitting, as evidenced by lower validation loss and higher validation metrics (accuracy, precision, recall, and F1-score) compared to higher learning rates. Furthermore, the gradual optimization enabled by lower learning rates allows the model to benefit from longer training durations (e.g., 8 or 18 epochs), refining its performance without plateauing or diverging. This makes lower learning rates particularly effective for nuanced tasks requiring high precision and generalization.

Lower batch sizes, such as 16, tend to perform better in the top 4 models because they allow the model to capture more detailed gradients during training, which is particularly advantageous in tasks like hate speech detection involving nuanced and diverse data. With smaller batches, the model processes fewer samples at a time, enabling it to better adapt to subtle patterns in the data, such as code-mixed contexts or cultural nuances. This precision helps reduce the risk of oversmoothing the gradients, which can occur with larger batch sizes like 32, where updates may generalize too broadly and miss finer details. Additionally, smaller batch sizes improve generalization, as the model sees a wider range of gradient variations during training, which is reflected in the lower validation loss and consistently higher metrics (accuracy, precision, recall, and F1-score) observed for batch size 16 in the top-performing models. These benefits make lower batch sizes more suitable for fine-tuned hate speech detection tasks where context sensitivity is critical.

The model was evaluated on a dedicated test set, comprising 10% portion of the whole dataset. The model achieved a near-perfect accuracy of 99.60%, false positive rate of 0.67% and false negative rate of 0.136%, reflecting high rate of correct predictions. With a precision of 99.60%, the model effectively minimized false positive predictions, while a recall of 99.60% exhibited the model capability on reducing false negatives. Additionally, the F1-score of 99.60% further demonstrates the model’s balance between precision and recall which indicates excellent performance on both types of classification errors. Figure 10 presents the confusion matrix generated from the test set inference results.

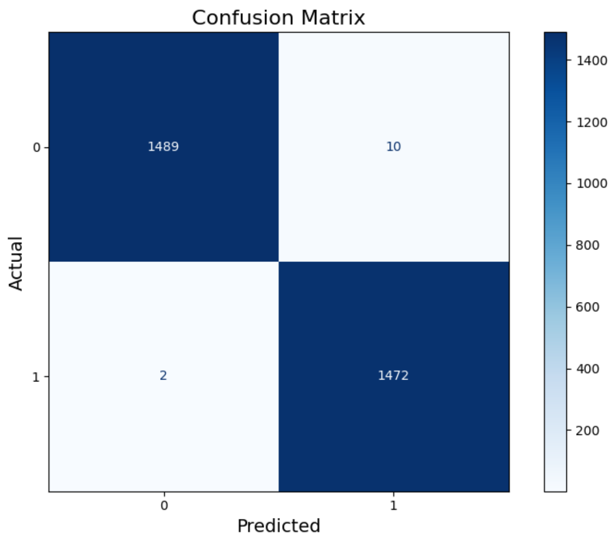


Figure 10: Primary test set confusion matrix.

Upon testing the supplementary test set, the model exhibited excellent performance, achieving 91.00% accuracy, 92.40% precision, 91.00% recall, and an F1-score of 90.94%. The false positive rate is approximately 17.6%, while the false negative rate is 0%. These results indicates that the trained model effectively manages outlier cases that reflect real-life scenarios, where sentence structures may be unpredictable and vary widely. Figure 11 shows the confusion matrix from the supplementary test set.

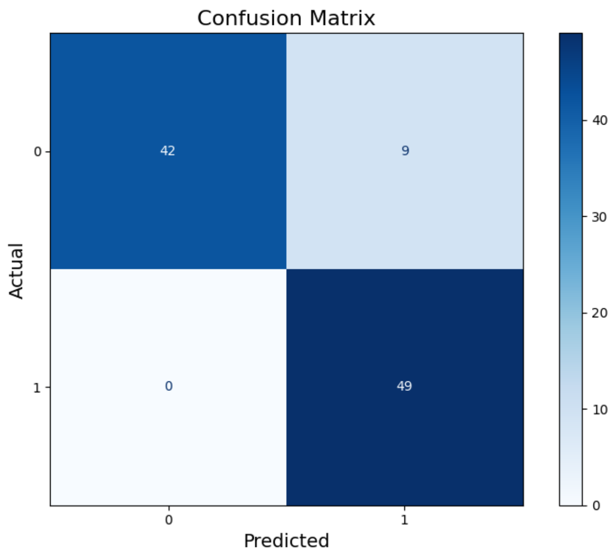


Figure 11: Supplementary test set confusion matrix.

Table 4 shows the false predictions from the model. The observed false positives in the model can be attributed to several factors that affect the model's ability to accurately classify sentiment. Firstly, the abbreviation of words that sometimes include number which are common in informal communication (e.g., "klo" for "kalau" and “bisa2” for “bisa-bisa”), may not consistently align with the model's learned patterns, leading to misinterpretations.

Table 4: Primary test set false prediction samples

Text	**Type**
"this health advice in the comments is really practical, informasi yang berguna!	False Positive
"organizing a book club in a cozy cafe promotes community, mempromosikan komunitas!"	False Positive
"golfers di sini terlalu angkuh, mental miskin!"	False Positive
"emission control technology is state-of-the-art, teknologi kontrol emisi sangat canggih."	False Positive
"recommending the park's features to friends promotes information_technology usage, mendorong penggunaannya!"	False Positive
"menjelajahi stasiun luar angkasa itu feels like angstrom real adventure, sungguh menakjubkan!"	False Positive
"the noise from aircraft itu unbearable, they should ban them from cities!"	False Positive
"every time a pet is brought here, it's a gamble with their health, berjudi dengan kesehatan hewan!"	False Positive
"great command! your tactics are impressive, komando yang hebat! taktikmu mengesankan!"	False Positive
"the observatory fails to engage with the global scientific community, gagal berinteraksi dengan komunitas ilmiah global!"	False Positive
"why are you indians still here if you don't support the cause? kenapa masih di sini?"	False Negative
"exploring this aquarium is never educational and fun, dasar jewish!"	False Negative

Furthermore, the presence of unknown keywords and emojis, such as internet slang or culturally specific terms (e.g., "wkwk"), which the model was not trained on, may contribute to confusion during classification. Lastly, the ambiguity of certain words—like "sumpah" (swear), "kelewatan" (gone too far), and phrases like "I cannot stand"—which may carry negative connotations but do not explicitly convey hate speech, poses challenges for the model. These are the reasons that likely caused slight performance evaluation drop on the supplementary test. These components also highlight the complexities involved in accurately interpreting code-mixed sentiment, underscoring the need for improved training data and contextual understanding by wider keyword enrichment including outlier vocabularies.

Table 5: Supplementary test set false prediction samples.

Text	**Type**
"Nggak tau sih soalnya lucu juga ya klo dipikir2 wkwk"	False Positive
"Aduh cape banget gw kerja sm orang sumpah"	False Positive
"Kenapa ya orang-orang tu bisa2nya jahat bgt?"	False Positive
"Gak mau jadi politikus gue; tekanannya psti gede bgt! 😵 😵"	False Positive
"Sumpah kesel bgt gw sama temen lo!"	False Positive
"Kamu mau gak jadi pacar aku?"	False Positive
Apakah kamu bidadari? Soalnya cantiknya kelewatan 😍😍😍😍"	False Positive
"I can't stand harga dri produk ini; mahal bgt! 😡"	False Positive
"Kerja kerasmu tidak akan sia-sia, keep going! 💪"	False Positive

Table 5 shows false positive predictions on the supplementary test set where the model incorrectly classified non-hateful texts as hate speech. The examples contain informal language, humor, slang, emojis, and exaggerations, which may have caused misclassification. The model likely struggles with contextual understanding, misinterpreting harmless expressions as offensive.

False positives and false negatives in hate speech detection have critical implications, both ethically and practically. False positives occur when non-hateful content is incorrectly classified as hate speech. This can suppress legitimate expression, create a chilling effect on free speech, and harm users who may feel unfairly censored or misjudged. Conversely, false negatives, where actual hate speech is not detected, allow harmful content to persist. This can perpetuate harm to targeted individuals or communities and undermine trust in the detection system. Failure to address false negatives can have serious societal impacts, such as the normalization of offensive language or inadequate protection for marginalized groups.

We apply several strategies in the hate speech detection system to mitigate the errors. First, enhancing the quality of training data is essential. This involves enriching datasets with diverse examples of internet slang, culturally specific terms, and nuanced expressions to improve the model's ability to capture contextual subtleties on real world examples. Second, employing context-aware models or fine-tuning pre-trained models like XLM-RoBERTa with additional layers designed for better contextual understanding can significantly improve classification accuracy. These strategies collectively enhance the robustness, fairness, and reliability of hate speech detection systems in practical applications. Table 6 shows common errors in hate speech detection along with examples and potential causes

Table 6: Common errors in hate speech detection.

Category	Example	Potential Cause
Lexical Ambiguity	*"sumpah"* (swear)	The model struggles to differentiate context-specific meanings without additional contextual cues.
Cultural Nuances	*"wkwk"* (Indonesia's slang for laughter) *"ashiap"* (Indonesia's slang for saying *"yes"*)	Limited representation of culturally specific terms in the training dataset, leading to misclassification.
Implicit Hate Speech	*"Gak mungkin banget sih someone from* ***** *bisa sepinter itu"*	The model struggles with identifying hate speech when explicit offensive keywords are absent.
False Positive	*"I don't agree, tapi nggak apa-apa sih."*	Neutral or positive statements incorrectly classified as hate speech due to negative sentiment keywords.
False Negative	*"Go back to your own country!"*	Inadequate coverage of explicit hate speech examples or poor generalization from training data.
Out-of-Vocabulary Terms	*"Sumpah itu tadi orang noob banget."* *"Bruh konser tadi, absolutely lit sih bro"* *"Cmon fam!"*	The model's tokenizer or vocabulary does not include these terms, leading to incomplete representation.

We assessed the generalization capability of our fine-tuned XLM-RoBERTa model for mixed-code hate speech detection using a post-hoc 5-fold stratified cross-validation strategy, with the best-performing model fine-tuned at a learning rate of 2e-5 and a batch size of 16. The dataset was divided into five stratified folds, ensuring balanced representation of hate speech and non-hate speech instances across each split. For each fold, the model was evaluated on the held-out validation set without further training, utilizing the Hugging Face Transformers library. The model achieved an average evaluation loss of 0.0342, with minimal variation across folds (ranging from 0.0322 to 0.0356), indicating strong and consistent performance.

We quantified the uncertainty in the model's generalization performance by computing a 95% confidence interval (CI) for the mean evaluation loss. This CI provides a range in which the true mean loss is expected to fall with 95% confidence. Applying the standard normal approximation method, we obtained a 95% CI of (0.0327, 0.0356). The narrow interval suggests that the model's performance is statistically stable, with low variance across different validation sets. The small margin of error highlights the model's high reliability, confirming its strong generalization ability and robustness to minor variations in the dataset. These results further emphasize the effectiveness of the fine-tuned XLM-RoBERTa model for mixed-code hate speech detection.

The results indicate that XLM-RoBERTa achieves exceptional performance when properly fine-tuned and trained with a substantial number of tokens. The model consistently outperforms those used in related studies across multiple multilingual hate speech detection tasks. As shown in Table 1, prior implementations of XLM-RoBERTa and its variations yielded F1-scores ranging from 32% to 92.55%, depending on the dataset and language pair. In contrast, the proposed model achieves an F1-score of 99.60% on the primary test set and 90.94% on the supplementary test set. Furthermore, while earlier studies reported relatively low precision and recall, the model maintains high precision (99.60% and 92.40%) and recall (99.60% and 91.00%), ensuring balanced classification performance. Error analysis reveals that the model effectively minimizes false negatives, with a false negative rate of 0.136% on the primary test set and 0% on the supplementary test set. Given these results, the model not only surpasses previous approaches in overall performance but also demonstrates robustness in handling linguistic variations and outlier cases, making it highly effective for mixed-code hate speech detection. Future research on multilingual hate speech detection could further benefit from incorporating additional training data tailored to specific language requirements.

6 Conclusion

This study demonstrates the effectiveness of XLM-RoBERTa in detecting hate speech within code-mixed texts, particularly in Indonesian-English code-mixed contexts. The model achieved a high level of accuracy (99.6%) on the primary test set and maintained strong generalization across realistic supplementary data on 91% accuracy, reflecting its robustness in handling varied linguistic inputs. These results highlight the importance of multilingual adaptability in hate speech detection, particularly for complex online environments where language boundaries are fluid. Future research could enhance these outcomes by incorporating additional real-world linguistic variations and expanding to other language pairs, contributing to safer and more inclusive digital spaces on broader language scopes.

Acknowledgement

The authors acknowledge the support of Universitas Multimedia Nusantara for this study and appreciate the constructive feedback provided by the reviewers and editors, which significantly contributed to improving the quality of this paper.

References

[1] J. B. Walther, "Social media and online hate," *Curr. Opin. Psychol.*, vol. 45, no. January, 2022, doi: 10.1016/j.copsyc.2021.12.010.

[2] K. Sreelakshmi, B. Premjith, and K. P. Soman, "Detection of Hate Speech Text in Hindi-English Code-mixed Data," *Procedia Comput. Sci.*, vol. 171, no. 2019, pp. 737–744, 2020, doi: 10.1016/j.procs.2020.04.080.

[3] A. Wicaksana, K. Sorensen, and F. Dinarta, "Enhancing Hate Speech Detection in Mixed-Language Texts: A Comparative Study of BLOOM and XLM-RoBERTa Models," in *2025 17th International Conference on Computer and Automation Engineering (ICCAE)*, IEEE, 2025.

[4] M. Terblanche, K. Olaleye, and V. Marivate, "Prompting Towards Alleviating Code-Switched Data Scarcity in Under-Resourced Languages with GPT as a Pivot," *3rd Annu. Meet. ELRA-ISCA Spec. Interes. Gr. Under-Resourced Lang. SIGUL 2024 Lr. 2024 - Work. Proc.*, pp. 272–282, 2024.

[5] J. S. Malik, H. Qiao, G. Pang, and A. van den Hengel, "Deep Learning for Hate Speech Detection: A Comparative Study," pp. 1–18, 2022, [Online]. Available: http://arxiv.org/abs/2202.09517

[6] T. Tita, Q. Mary, and A. Zubiaga, "Cross-lingual Hate Speech Detection using Transformer Models", doi: 10.48550/arXiv.2111.00981.

[7] S. Wang, J. Liu, X. Ouyang, and Y. Sun, "Galileo at SemEval-2020 Task 12: Multi-lingual Learning for Offensive Language Identification using Pre-trained Language Models," *14th Int. Work. Semant. Eval. SemEval 2020 - co-located 28th Int. Conf. Comput. Linguist. COLING 2020, Proc.*, pp. 1448–1455, 2020, doi: 10.18653/v1/2020.semeval-1.189.

[8] L. B. Hutama and D. Suhartono, "Indonesian Hoax News Classification with Multilingual Transformer Model and BERTopic," *Inform.*, vol. 46, no. 8, pp. 81–90, 2022, doi: 10.31449/inf.v46i8.4336.

[9] A. Conneau *et al.*, "Unsupervised Cross-lingual Representation Learning at Scale." [Online]. Available: https://github.com/facebookresearch/cc

[10] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.11692

[11] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference,* 2019.

[12] A. Conneau *et al.*, "Unsupervised Cross-lingual Representation Learning at Scale," Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.02116

[13] Y. Meng *et al.*, "Representation Deficiency in Masked Language Modeling," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.02060

[14] X. Amatriain, A. Sankar, J. Bing, P. K. Bodigutla, T. J. Hazen, and M. Kazi, "Transformer models: an introduction and catalog," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.07730

[15] G. Lample and A. Conneau, "Cross-lingual Language Model Pretraining," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.07291