Scientific Journal of Informatics
Vol. 13, No. 3, August 2026
p-ISSN 2407-7658 https://journal.unnes.ac.id/journals/sji/index e-ISSN 2460-0040

An Ablation Study on Stacked LSTM and SMOTE for Government Policy Sentiment Classification

Sima Maulina[1*], Catur Edi Widodo[2], Budi Warsito[3]
[1]Master's Program in Information Systems, Postgraduate School, Diponegoro University, Indonesia
[2]Department of Physics, Faculty of Sciences and Mathematics, Diponegoro University, Indonesia
[3]Department of Statistics, Faculty of Sciences and Mathematics, Diponegoro University, Indonesia

Abstract.
Purpose: We compare four deep learning configurations of Standard LSTM without SMOTE, Standard LSTM with SMOTE, Stacked LSTM without SMOTE, and Stacked LSTM with SMOTE used for multi-class sentiment classification of Indonesian public comments on the government's Free Nutritious Meals Program (MBG) policy posted on Instagram. Most existing research on this topic relies on single-layer LSTM models that fail to address class imbalance and the layered, often sarcastic semantic structure commonly found in informal Indonesian social media writing at the same time. We test whether a Stacked LSTM architecture combined with SMOTE can simultaneously model sequential word dependencies and high-level semantic patterns, while correcting the bias that class imbalance tends to introduce.
Methods: We collected 1,000 Instagram comments from the account @ferryirwandi using the Instaloader library and labeled them into three sentiment classes with TextBlob. Preprocessing was carried out through six steps: cleaning, case folding, slang normalization with a custom dictionary, stopword removal, tokenization, and stemming via Sastrawi after which TF-IDF was used for feature extraction. The data was split 80:20 into training and testing sets before applying any oversampling; SMOTE was applied only to the training data, so the test set remained intact and free of data leakage. Performance was measured using Accuracy, Macro-Precision, Macro-Recall, and Macro-F1 Score, obtained from a multiclass confusion matrix based on a One-vs-Rest scheme.
Result: Scenario 3 (Stacked LSTM without SMOTE) produced the top result, reaching 95.00% Accuracy, Macro-Precision of 0.9276, Macro-Recall of 0.8752, and Macro-F1 Score of 0.8973 the best outcome among the four configurations. In contrast, Scenario 4 (Stacked LSTM with SMOTE) performed worse, with accuracy dropping to 92.50% and the F1-Score falling to 0.8178. We attribute this decline to feature space overlap: the synthetic minority samples generated by SMOTE landed too close to the dominant neutral class within the high-dimensional, sparse TF-IDF feature space.
Novelty: Our findings provide empirical support for the idea that hierarchical representations built by stacked recurrent layers are well suited to capturing the complex context, sarcasm, and long-range semantic relationships found in Indonesian political discourse. We also propose the four-scenario ablation design as a reusable method for disentangling how much architectural depth and data balancing each contribute, both independently and jointly, when working with imbalanced Indonesian social media sentiment data.
Practical Implications: For practitioners working with TF-IDF features on narrow-topic datasets, our findings suggest that Stacked LSTM without oversampling is the more reliable configuration for multi-class sentiment analysis of Indonesian government-policy discourse on social media.

Keywords: : Sentiment Analysis, Government Policy, Stacked LSTM, SMOTE, Class Imbalanced
Received June 2026 / **Revised** July 2026 / **Accepted** July 2026

INTRODUCTION

Social media platforms such as Instagram have moved well beyond their original role as photo- and video-sharing applications [1]. Today they function as public arenas where citizens openly discuss social and political issues [2]. Governments and public figures have recognized this, using these platforms as two-way communication channels, with comment sections in particular turning into a live, ongoing record of public feedback that blends constructive criticism, policy suggestions, and outright public support [3].

*Corresponding author.
Email addresses: simamaulina@students.undip.ac. (Maulina)*
DOI: 10.15294/sji.v13i3.53853

One policy currently triggering intense, polarized discussion is the government's Makan Bergizi Gratis (MBG) program [4]. Manually reading through thousands of comments to gauge public opinion is slow and prone to the researcher's own subjectivity [5]. This is exactly the kind of problem NLP and deep learning are suited to solve, since both enable faster, more objective evaluation at scale [6], [7]. However, Indonesian social media text comes with its own complications: comments are unstructured, full of informal slang, and rarely balanced across sentiment categories [8]. When this imbalance is left unaddressed, models tend to become very good at predicting whichever class dominates the dataset while almost entirely missing the minority classes and it is often precisely there that the most useful public insight ends up hidden or overlooked .

To address this problem, we built a deep learning pipeline based on Stacked Long Short-Term Memory (Stacked LSTM) combined with the Synthetic Minority Over-sampling Technique (SMOTE). A Stacked LSTM differs from the standard single-layer version by stacking multiple recurrent layers on top of one another, allowing the network to extract textual features at different levels of abstraction [9]. The lower layer captures basic sequential word patterns while the layers above it build increasingly abstract semantic representations, this is what allows the model to detect things like irony, long-range context, and subtle emotional shifts [8]. Alongside this architecture, SMOTE rebalances the training data by generating synthetic samples for underrepresented classes through linear interpolation in feature space [1].

Naive Bayes, Decision Trees, and SVM have long been popular for sentiment classification because they are easy to implement and interpret [10]. Their weakness lies in their dependence on hand-crafted features and their inability to genuinely capture sequential dependencies or long-range relationships in text [11], [12]. This limitation is even more pronounced in Indonesian social media writing, where slang, code-mixing, and sarcasm demand a level of semantic understanding that goes beyond simple token matching [13]. LSTM-based architectures, on the other hand, have shown a clear advantage in sentiment classification tasks precisely because they are designed to model temporal dependencies in sequential data [14].

Research by Arwindarti et al. [4] and Novfuja et al. [15] has shown that LSTM is effective for sentiment classification in Indonesian public-policy discourse. However, both studies rely exclusively on single-layer LSTM architectures, which carry a structural constraint that limits how deeply they can model text: without multiple stacked recurrent layers, these models can only capture first-order sequential dependencies between adjacent tokens, and are therefore unable to extract the hierarchical, high-level semantic abstractions such as sarcasm detection, complex metaphor interpretation, and long-range contextual coherence that informal Indonesian social media writing frequently demands. This architectural ceiling becomes particularly acute when the input features are sparse, high-dimensional TF-IDF vectors, where the representational burden placed on a single recurrent layer is already high. Stacking LSTM layers vertically addresses this structural limitation by allowing the network to progressively build higher-order feature abstractions across layers [16]: the first layer captures low-level token sequences, while each subsequent layer re-encodes those representations at a more abstract semantic level. Younas et al. [9] confirmed that Stacked LSTM consistently outperforms single-layer LSTM in detecting contextual emotional correlations, even achieving micro F1-scores comparable to Transformer-based models at a substantially lower computational cost.

Class imbalance is a separate problem in its own right. When one class say, the neutral class dominates a dataset, standard classifiers tend to develop a strong bias toward it, which can push minority-class recall down close to zero [17]. SMOTE addresses this by generating new synthetic points through k-nearest-neighbor interpolation in feature space, pushing the model toward broader decision boundaries [18]. Aditya et al. [17] reported a 9.69% average accuracy improvement from combining SMOTE with LSTM in a study using a presidential-election sentiment dataset. What remains less clear, however, is how SMOTE interacts with a Stacked LSTM architecture specifically when the feature space is the high-dimensional, sparse kind produced by TF-IDF this combination has not been widely studied.

Most existing research has examined single-layer LSTM or basic oversampling separately rather than together [19]. Very few studies investigate how the interaction between stacked recurrent depth and SMOTE-based balancing plays out for multi-class Indonesian political sentiment, particularly for government programs promoted through educational influencer accounts. This is the gap we aim to close with an ablation study that isolates and evaluates each component's contribution separately. Our objectives

are: (1) to compare Standard LSTM against Stacked LSTM across four scenarios, with and without SMOTE, and (2) to measure how architectural depth and data balancing both separately and combined affect Accuracy, Macro-Precision, Macro-Recall, and Macro-F1 Score in multi-class Indonesian sentiment classification [4].

METHODS

Our experimental framework proceeds through nine sequential stages, chosen to keep the process replicable and methodologically sound. Beginning with raw data collection through web scraping, the workflow continues through labeling, text preprocessing, TF-IDF feature extraction, data splitting, SMOTE-based class balancing, model construction across the four ablation scenarios, multiclass confusion matrix evaluation, and finally visualization. Each stage was carried out in sequence to ensure a fair comparison across configurations; whenever a model's performance fell short of an acceptable threshold, we revisited the hyperparameters before proceeding to final evaluation. Figure 1 shows the complete workflow.

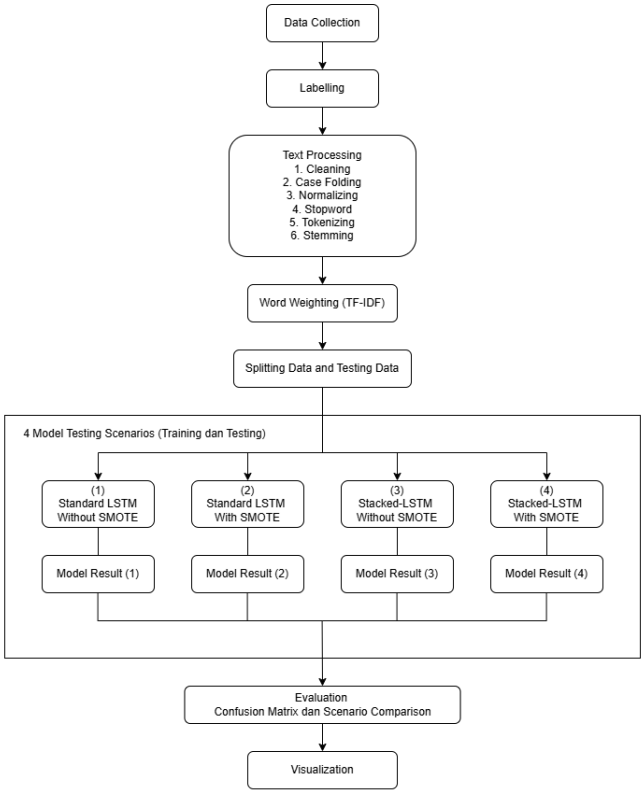

Figure 1. Flowchart of the proposed research procedure

Data Collection and Labeling

We built the primary dataset by extracting a single Instagram post from the educational political commentator @ferryirwandi that specifically discussed the government's Makan Bergizi Gratis (MBG) program, using Python's Instaloader library. We selected this account for its reach roughly 4.6 million followers which makes its comment section a fairly representative sample of organic public opinion on Indonesian government policy [4]. We limited the corpus to the first 1,000 comments posted within 14 days of the original post, based on the assumption, supported by earlier research, that discussion tends to be most active and most genuinely responsive within that early window, while later comments tend to be secondary reactions less tied to the initial policy news.

Each record in the raw dataset contained three fields: username, timestamp, and comment text. We grouped comments into three sentiment categories Positive, Negative, and Neutral using TextBlob's lexicon-based polarity scoring to automate labeling at scale [20]. The resulting distribution was heavily skewed: the Neutral class dominated, followed by Negative, with Positive making up only a small minority [21]. This

imbalance is the central problem SMOTE is meant to address in this study. Table 1 presents the dataset's characteristics and class distribution prior to oversampling.

Text Processing

How well a model performs downstream depends heavily on the quality of preprocessing applied before feature extraction, this matters especially for Indonesian social media text, given how far it deviates from formal language through abbreviations, regional slang, and mixed-language phrasing [18]. We processed the raw corpus through six preprocessing steps to remove noise and standardize informal input before extracting features; Table 2 shows annotated examples for each step. (1) Cleaning removed all non-textual elements: HTML tags, URLs, @-mentions, hashtags, numbers, emojis, and punctuation reducing dimensionality and eliminating tokens that carry no sentiment signal. The comment "Ini bagus banget!!! Cek info di @kemenkeu #MBG," for instance, becomes "Ini bagus banget Cek info di" after this step. (2) Case folding converted all letters to lowercase so the model would not treat "Pemerintah" and "pemerintah" as two different tokens. (3) Normalization mapped informal abbreviations and slang to their standard Indonesian forms using a dictionary built specifically for this corpus "ga" becomes "tidak", "bgt" becomes "banget", "yg" becomes "yang", and so on. This step matters a great deal, since informal Indonesian text is full of non-standard word forms that would otherwise be treated as out-of-vocabulary noise. (4) Stopword removal filtered out high-frequency function words with low sentiment weight conjunctions such as "dan" and "atau", prepositions such as "di", "ke", and "dari", and other filler words narrowing the feature space down to words that can actually discriminate between sentiment categories. (5) Tokenization split sentences into individual word units for frequency analysis and vectorization, which made the subsequent word-weighting stage easier to carry out. (6) Stemming reduced inflected words to their root form using the Sastrawi library, which implements the Enhanced Confix Stripping (ECS) algorithm for Bahasa Indonesia. Words such as "mendukung" and "didukung", for example, are both reduced to "dukung", keeping semantically related variants consistent [15].

Word Weighting (TF-IDF)

After preprocessing, we converted the cleaned token sequences into numerical vectors using Term Frequency–Inverse Document Frequency (TF-IDF), a weighting scheme that scores how important a term is to a given document relative to the entire corpus. Terms that appear frequently in a specific document but rarely elsewhere are assigned greater weight, which is useful for identifying sentiment-bearing keywords [22]. We chose TF-IDF over contextual embeddings such as Word2Vec or IndoBERT for two reasons. First, this method is computationally light and does not require GPU resources, which suits a dataset of 1,000 comments a fairly modest size [23]. Second, because the entire corpus centers on a single topic (the MBG policy), TF-IDF term-weighting tends to effectively isolate policy-specific jargon, local slang, and sentiment vocabulary, without the overfitting risk that comes with high-dimensional contextual embeddings on a narrow-domain dataset like this one [15]. Word weights were calculated using the equations below:

$$tf_{t,d} = \begin{cases} 1 + log_{10} tf_{t,d}, if tf_{t,d} > 0 \\ 0 \qquad otherwise \end{cases} \tag{1}$$

$$idf_t = log_{10}\left(\frac{N}{df_t}\right) \tag{2}$$

$$W_{t,d} = W_{tft,d} \times idf_t \tag{3}$$

Where N denotes the total number of documents in the corpus, and 〖df〗_t represents the specific document frequency containing the target term t.

Data Splitting and Class Imbalanced Handling (SMOTE)

Before performing any oversampling, we shuffled the entire 1,000-comment TF-IDF vector space to remove ordering bias, then split it 80:20 into a training set (800 samples) and a testing set (200 samples). This split was performed first, before any data augmentation which matters, because applying SMOTE before splitting risks data leakage, where synthetic samples drawn from the test distribution end up contaminating training and artificially inflating performance on held-out data [17]. By keeping the 200-sample test set in its original, imbalanced state, we ensure the evaluation results reflect how the model would actually perform on real, naturally distributed data. This same test set was used unchanged across all

four scenarios, keeping comparisons between them fair. SMOTE was then applied only to the 800-sample training partition. Rather than simply duplicating existing minority-class samples which risks overfitting SMOTE constructs entirely new data points through k-nearest-neighbor linear interpolation in feature space. For a given minority-class sample x_i, the technique identifies its k nearest neighbors and generates a new synthetic sample xnew at a point along the line connecting x_i to one of those neighbors, xzi, chosen at random. This pushes the classifier toward broader, more generalizable decision boundaries for underrepresented classes. The exact formulation is given in Equation (4), and Table 3 summarizes the full data-preparation workflow, including the split ratio and where SMOTE was applied.

$$x_{new} = x_i + \lambda \times (x_{zi} - x_i) \quad (4)$$

Where xnew is the newly generated synthetic sample, xi is the selected minority class feature vector, xzi is one of its k-nearest neighbors in the feature space, and λ is a random number drawn uniformly between 0 and 1 that determines where the new point falls along that line [17].

Stacked LSTM Architectural Design

Our proposed architecture is a Stacked Long Short-Term Memory (Stacked LSTM) network essentially a standard single-layer LSTM extended by stacking two or more recurrent layers vertically within a single sequential model. The reasoning behind this lies in hierarchical feature abstraction. In a standard single-layer LSTM, the hidden state from the final timestep feeds directly into the classification layer, which limits the model to capturing only first-order sequential relationships between tokens [24]. Stacking layers changes this: the lower layer extracts low-level sequential word patterns, while the upper layer takes the full hidden-state sequence from the first layer as input to build progressively more abstract, higher-level semantic representations. This kind of layered processing turns out to be essential for informal Indonesian social media text, which is full of complex metaphors, inverted sentence structures, layered sarcasm, and sentiment-modifying particles that require understanding context at more than one scale to classify correctly [9], [13]. We converted the 2D TF-IDF feature matrices into dense arrays and reshaped them into 3D tensors formatted as [samples, timesteps = 1, features], matching the 3D tensor input expected by Keras's LSTM layer, by treating each comment as a single-timestep sequence containing all of its TF-IDF feature dimensions. This allows the LSTM to process each comment as a temporal sequence while still preserving the full feature vector. The Stacked LSTM model itself combines two recurrent layers with the following parameters.

1. **First LSTM Layer:** Processes the baseline input sequences, configured with return_sequences=True so the network outputs its hidden states for every timestep to the layer above it. It is bounded by a Dropout rate of 0.3.
2. **Second LSTM Layer:** Extracts higher-level semantic features, configured with return_sequences=False to compress the sequential outputs into a single vector for final classification. It also applies a 0.3 Dropout boundary.
3. **Dense Output Layer:** Three fully connected neural nodes representing the Positive, Negative, and Neutral classes, using the Softmax activation function to map outputs into a final probability distribution.

We trained the models using the Adam Optimizer with a fixed learning rate of 0.001, Categorical Crossentropy for multi-class loss, a batch size of 32, and Early Stopping to prevent overfitting. Figure 2 details the network's layer configuration and data flow.

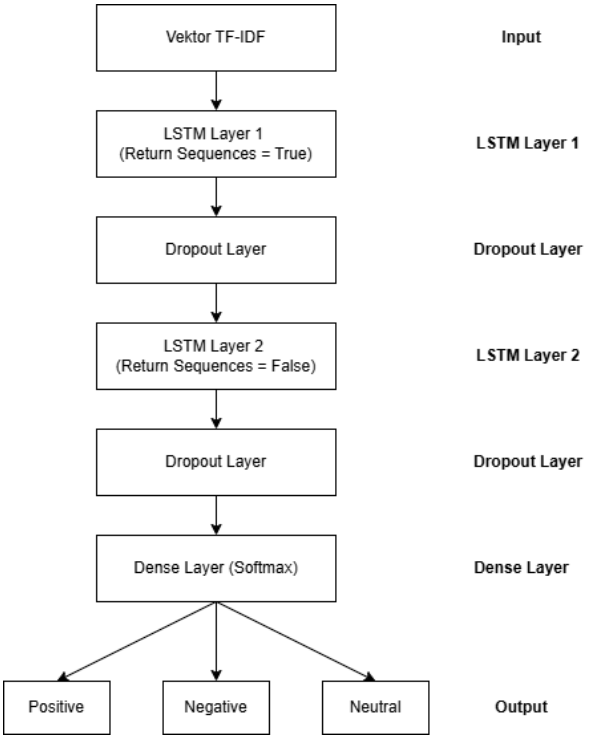

Figure 2. Architecture diagram of the proposed stacked lstm network

Model Training and Hyperparameter Configuration

We trained all four scenarios under identical hyperparameters, so that any performance differences observed can be attributed to architecture or data preparation rather than training variance. Adam was chosen as the optimizer because its adaptive learning rate adjusts gradient updates independently for each parameter, a property well suited to the sparse, high-dimensional TF-IDF matrices typical of text classification, where gradients tend to be irregular across feature dimensions [16]. We set a fixed learning rate of 0.001 as a middle ground that avoids overshooting the gradient while still converging efficiently. Categorical Crossentropy was a natural choice of loss function here, since it is designed for mutually exclusive multi-class problems and penalizes incorrect probability distributions in proportion to how wrong they are, which makes it sensitive to misclassification across our three sentiment categories. For batch size, we chose 32, which provided efficient gradient updates without the noise that smaller batches tend to introduce. We allowed up to 100 training epochs, giving the Stacked LSTM's hierarchical layers enough room to converge on complex sequential patterns, and used Early Stopping with a patience of 10 epochs to automatically halt training once validation loss stopped improving, restoring the best-performing weights at that point. The 0.3 Dropout rate applied after each LSTM layer follows the empirical results of Musthafa et al. [16], who found that this rate suppresses overfitting in Stacked LSTM down to as low as 0.24% without sacrificing accuracy. We fixed the same random seed across NumPy and TensorFlow to ensure reproducibility. Table 4 lists the complete hyperparameter summary.

Model Evaluation

To evaluate the four scenarios, we used a Multiclass Confusion Matrix with a One-vs-Rest decomposition, in which each of the three sentiment classes (Positive, Negative, Neutral) is treated, in turn, as a binary problem against the other two classes, yielding per-class True Positive, True Negative, False Positive, and False Negative counts. From this data, we calculated four aggregate metrics. Accuracy is simply the percentage of correctly classified samples out of the total. Macro-Precision averages, across classes, the proportion of predicted-positive samples that were actually correct. Macro-Recall averages, across classes, how many of the actual positive samples in each class the model successfully detected; this is the metric that reflects sensitivity per sentiment category. Macro-F1 is the harmonic mean of per-class precision and recall, providing a balanced measure that remains valid even under class imbalance. We chose Macro-Average over Micro-Average or Weighted-Average specifically because it weighs all three classes equally regardless of how many samples each class has, which matters here, since a Weighted- or Micro-Average would let the dominant Neutral class mask how poorly the model handles the minority Positive class. This

keeps our reported metrics honest about performance across every sentiment category, consistent with standard practice for imbalanced text classification [12], [21].

The four evaluation metrics are formally defined by Equations (5) through (8) below. Let TP denote True Positives (samples correctly classified as belonging to a given class), TN denote True Negatives (samples correctly classified as not belonging to that class), FP denote False Positives (samples incorrectly assigned to the class), FN denote False Negatives (samples belonging to the class that the model missed), and C denote the total number of classes (C = 3 in this study):

$$Accuracy = \frac{TP \times TN}{TP + TN + FP + FN} \times 100\% \quad (5)$$

$$Macro - Precision = \frac{1}{C} \sum_{i=1}^{C} \frac{TP_i}{TP_i + FP_i} \quad (6)$$

$$Macro - Recall = \frac{1}{C} \sum_{i=1}^{C} \frac{TP_i}{TP_i + FN_i} \quad (7)$$

$$Macro - F1 = 2 \times \frac{Macro{-}Precision \times Macro{-}Recall}{Macro{-}Precision + Macro{-}Recall} \quad (8)$$

Equation (5) measures the overall proportion of correctly classified samples across all classes. Equations (6) and (7) compute per-class Precision and Recall independently, then average them arithmetically so that each class, regardless of its size, contributes equally to the final score. Equation (8) synthesizes those two averages into a single balanced measure that penalizes configurations that sacrifice one for the other.

Experimental Scenarios

To isolate the contribution of each architectural and data-preparation choice, we designed four scenarios as an ablation study. Each scenario introduces a single controlled change either to the architecture (single-layer LSTM vs. Stacked LSTM) or to the training data (with or without SMOTE) while holding all other factors constant. Scenario 1 (Standard LSTM Without SMOTE) is our baseline: a conventional single-layer LSTM trained on the original, imbalanced dataset. Scenario 2 (Standard LSTM with SMOTE) adds oversampling to that same single-layer architecture, allowing us to measure the effect of data balancing on its own. Scenario 3 (Stacked LSTM Without SMOTE) keeps the imbalanced data but replaces the architecture with the two-layer recurrent design, isolating the contribution of architectural depth on its own. Scenario 4 (Stacked LSTM with SMOTE) combines both interventions, and going in, we expected it to be the strongest configuration. Comparing across these four scenarios provides empirical evidence for how architectural depth and class balancing each affect both independently and jointly multi-class sentiment classification performance. Table 5 summarizes the four configurations.

RESULTS AND DISCUSSIONS

We trained all four ablation configurations in parallel over 100 epochs and computed their macro performance metrics. Global Accuracy, Macro-Precision, Macro-Recall, and Macro-F1 Score were all calculated against an untouched 200-row test set, using a multiclass confusion matrix.

Multiclass Confusion Matrix Analysis

Table 1 below presents the complete dataset characteristics and class distribution across all partitions, including the effects of SMOTE on the training set.

Table 1. Dataset characteristics and class distribution

Data Split	Positive	Negative	Neutral	Total
Full Dataset	90	345	565	1,000
Training Set (800)	72	276	452	800
Testing Set (200)	18	69	113	200
After SMOTE (Training only)	452	452	452	1,356

The figures confirm that the stratified 80:20 split maintained the original class proportions exactly in both partitions. After SMOTE, all three classes in the training set were brought to 452 samples each, eliminating the original 6.3:1 imbalance ratio between Neutral and Positive classes. The test set was left unchanged across all four scenarios to ensure a fair and uncontaminated evaluation baseline. Running each configuration against the 200 held-out validation samples consisting of 69 actual Negative, 113 actual Neutral, and 18 actual Positive samples produced the classification boundaries shown in Figure 3. The exact breakdown of correct predictions along the diagonal, along with where each model made errors, is laid out in Tables 2 through 5.

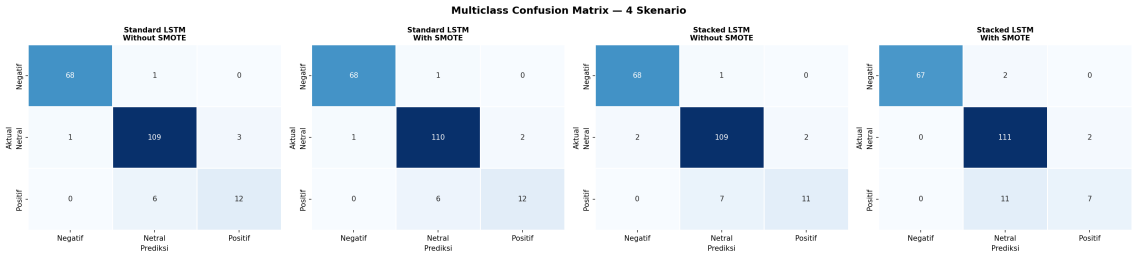

Figure 3. Multiclass confusion matrix heatmaps across the four experimental scenarios

Table 2. Confusion matrix for scenario 1 (standard LSTM without SMOTE)

	Prediction Classification			Total Actual
	Positive	**Negative**	**Neutral**	
Positive	10	0	8	18
Negative	0	68	1	69
Neutral	2	2	109	113

Table 3. Confusion matrix for scenario 2 (standard LSTM With SMOTE)

	Prediction Classification			Total Actual
	Positive	**Negative**	**Neutral**	
Positive	12	0	6	18
Negative	0	68	1	69
Neutral	3	1	109	113

Table 4. Confusion matrix for scenario 3 (sacked LSTM without SMOTE)

	Prediction Classification			Total Actual
	Positive	**Negative**	**Neutral**	
Positive	12	0	6	18
Negative	0	68	1	69
Neutral	2	1	110	113

Table 5. Confusion matrix for scenario 4 (stacked LSTM with SMOTE)

	Prediction Classification			Total Actual
	Positive	**Negative**	**Neutral**	
Positive	7	0	11	18
Negative	0	67	2	69
Neutral	1	1	111	113

Cross-Scenario Performance Comparison

Figure 4 shows the training and validation loss and accuracy curves for all four scenarios over the course of training. All four converge in a fairly consistent way, with training accuracy climbing quickly toward 100% while validation accuracy stabilizes between 92% and 97%. There is a visible gap between training loss (which approaches 0) and validation loss (which stabilizes around 0.2–0.3) in every scenario; this indicates a degree of overfitting, which is unsurprising given the relatively small dataset (1,000 samples) and the inherently sparse nature of TF-IDF feature vectors. Scenario 4 (Stacked LSTM with SMOTE) stands out for showing the largest fluctuation in validation accuracy early in training, consistent with it ultimately performing worst on the test set. Figure 5 visualizes the aggregate metrics as clustered bar charts, and Table 6 lists the exact figures.

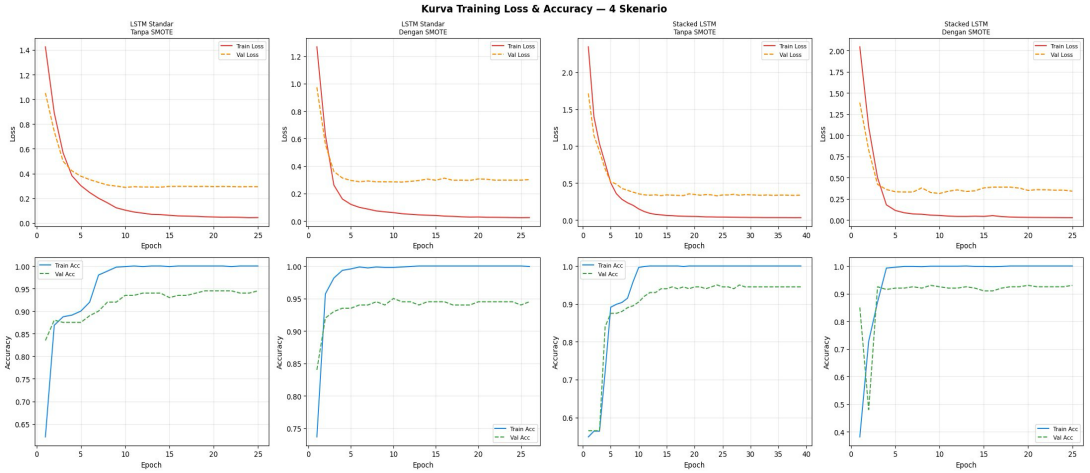

Figure 4. Training and validation loss & accuracy curves across four experimental scenarios

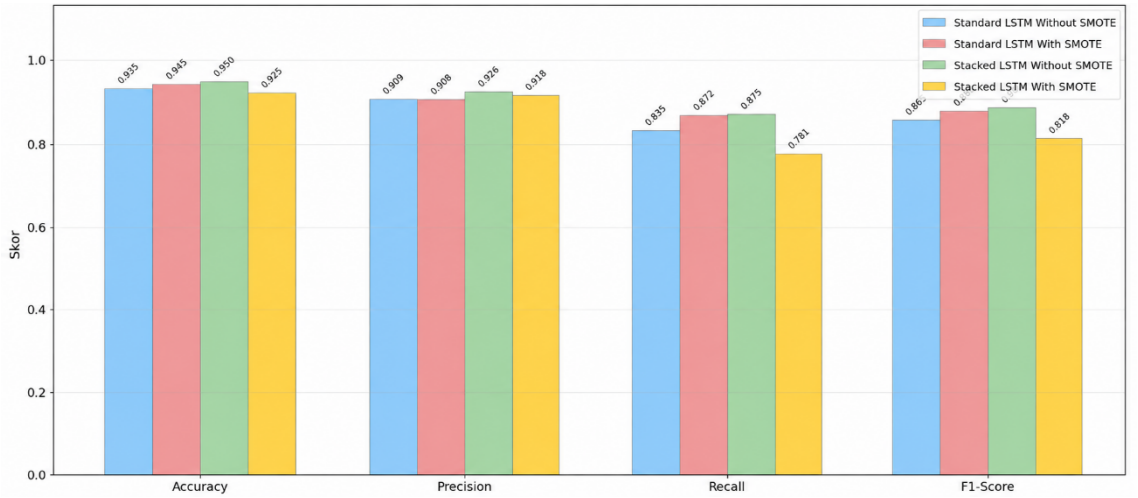

Figure 5. Clustered bar charts comparing performance metrics across all scenarios

Table 6. Consolidated performance evaluation summary matrix

No	Experimental Modeling Scenarios	Accuracy	Precision	Recall	F1-Score
1	Scenario 1 - Standard LSTM Without SMOTE	0.9350	0.9095	0.8352	0.8629
2	Scenario 2 - Standard LSTM With SMOTE	0.9450	0.9084	0.8723	0.8882
3	Scenario 3 - Stacked LSTM Without SMOTE	0.9500	0.9276	0.8752	0.8973
4	Scenario 4 - Stacked LSTM With SMOTE	0.9250	0.9185	0.7807	0.8178

Table 6 shows that the ablation framework successfully identified real and fairly substantial performance differences between configurations. Among the single-layer baselines, balancing the classes clearly helped: Scenario 2 (Standard LSTM with SMOTE) raised accuracy from 0.9350 to 0.9450 and lifted the Macro F1-Score by 2.53% (to 0.8882) compared to the unbalanced Scenario 1 baseline.

The biggest jump, however, came from changing the architecture itself in Scenario 3 (Stacked LSTM Without SMOTE). This configuration outperformed all others, reaching 95.00% accuracy and a Macro F1-Score of 0.8973 the best overall result. Its Macro-Precision reached 0.9276, backed by a solid Macro-Recall of 0.8752 [25]. By contrast, adding SMOTE to that same stacked architecture in Scenario 4 had the opposite effect: accuracy dropped to 0.9250 and F1 fell to 0.8178, the lowest score recorded in this study.

Theoretical Insights and Analytical Discussion

Scenario 3's strong result is consistent with what hierarchical representation theory predicts for sequence modeling [26]. The single-layer models in Scenarios 1 and 2 are limited to detecting relatively linear, syntactic-level token relationships, whereas Scenario 3's two-layer configuration allows the first recurrent

layer to capture low-level sequential structure, which the second layer then transforms into more abstract, higher-level semantic representations [9]. This layered approach proved highly influential in classifying the @ferryirwandi comment corpus, which is full of complex metaphors, inverted sentence structures, and the kind of internet sarcasm users deploy when debating the economic feasibility of the MBG budget [15].

The drop observed in Scenario 4 reveals a real limitation of oversampling when paired with deep learning architectures [16]. Out of 18 actual positive samples, Scenario 4 correctly identified only 7, while the other 11 were misclassified as neutral. This alone pulled the positive class's recall down to 0.39, which in turn dragged the overall Macro-Recall average down to 0.7807.

Feature space overlap is likely the best explanation for this. TF-IDF produces high-dimensional, sparse vectors in which the boundaries between sentiment categories are often thin and poorly defined to begin with. When SMOTE interpolates new synthetic positive samples, these new points can easily end up in territory already dominated by the neutral class [17]. This blurs the decision boundary the model needs to learn, leading to over-generalization and weaker macro performance once the model is tested on real test data.

Comparison with Previous Studies

To put our results in context, Table 7 compares our findings with prior studies that used similar deep learning approaches for Indonesian and general-purpose sentiment classification, covering architecture, dataset, language, SMOTE usage, and the best reported metrics [27].

Table 7. Comparative performance against prior studies

Study	Architecture	Dataset	Language	SMOTE	Accuracy	Macro F1
Arwindarti et al. [4]	Standard LSTM	Instagram Gov.	Indonesian	No	87.50%	0.8420
Aditya et al. [17]	LSTM + SMOTE	Twitter Election	Indonesian	Yes	91.00%	0.8800
Najwa et al. [28]	BiLSTM + B-SMOTE	Twitter Telco	Indonesian	Yes	91.20%	0.8730
Novfuja et al. [15]	Standard LSTM	Instagram MBG	Indonesian	No	89.30%	0.8610
Dubey et al. [25]	Deep LSTM	Product Reviews	English	No	93.50%	0.9200
This Study (S3)	Stacked LSTM	Instagram MBG	Indonesian	No	95.00%	0.8973
This Study (S4)	Stacked LSTM+SMOTE	Instagram MBG	Indonesian	Yes	92.50%	0.8178

The figures in Table 7 confirm that our Stacked LSTM (Scenario 3) achieves the highest accuracy 95.00% and a competitive Macro F1-Score of 0.8973 among the Indonesian social-media studies compared here. Arwindarti et al. [4] used single-layer LSTM for Instagram sentiment analysis and reported 87.50% accuracy; Aditya et al. [17] combined SMOTE with standard LSTM and achieved an F1-Score of 0.88; Najwa et al. [28] used BiLSTM with Borderline-SMOTE and reached 91.20%; Novfuja et al. [15], studying the same MBG topic with standard LSTM, reported 89.30%. Our Stacked LSTM without SMOTE outperforms all of these, which we attribute to its two-layer hierarchical structure, capable of capturing both low-level sequential token patterns and high-level abstract semantics at once.

This aligns with the broader pattern in the literature, where architectural depth tends to improve classification accuracy on informal social media text [25], and supports Musthafa et al.'s [16] finding that a properly regularized Stacked LSTM can suppress overfitting even on small datasets. Where our results diverge from prior work is in the effect of SMOTE in Scenario 4: Aditya et al. [17] reported that SMOTE improved standard LSTM performance, the opposite of what we found here. We argue that this discrepancy comes down to feature space: dense word embeddings give SMOTE a compact, semantically meaningful space in which to interpolate, whereas sparse TF-IDF matrices produce synthetic samples with far blurrier, more ambiguous boundaries that tend to land within the dominant neutral region.

CONCLUSION

This study set out to evaluate a Stacked LSTM model, with and without SMOTE, for multi-class sentiment classification of Indonesian public comments on the government's Makan Bergizi Gratis (MBG) policy on Instagram. Our ablation results show that Stacked LSTM without synthetic oversampling Scenario 3 delivered the best performance, reaching 95.00% accuracy and a Macro F1-Score of 0.8973. Stacking recurrent layers proved highly effective at handling the complex syntax and informal phrasing characteristic of Indonesian social media discourse.

By contrast, pairing SMOTE with high-dimensional TF-IDF vectors in Scenario 4 hurt performance. The oversampling process generated feature space overlap that blurred the classification boundaries, causing the model to misclassify positive comments as neutral. This is consistent with earlier literature indicating that SMOTE's effectiveness is strongly dependent on how dense and separable the underlying feature space is [17].

Several limitations deserve mention. The dataset covers only 1,000 comments from a single Instagram post, which may not fully represent the range of public opinion on the MBG policy across other platforms or demographic groups. Automatic labeling via TextBlob a lexicon-based tool designed primarily for English may have introduced labeling noise in the Indonesian text, despite the normalization preprocessing applied beforehand. TF-IDF, while computationally efficient, also lacks the ability to handle contextual word-sense disambiguation. Future work could address these limitations by exploring contextual pre-trained language models such as IndoBERT or RoBERTa-based Indonesian variants, which provide denser, more semantically rich feature spaces less susceptible to the sparse-interpolation problem identified here [29]. Expanding the dataset across multiple posts and platforms, together with cross-validation, would also help strengthen the generalizability of the findings [30].

REFERENCES

[1] P. Nandwani and R. Verma, "A review on sentiment analysis and emotion detection from text," *Soc. Netw. Anal. Min.*, vol. 11, no. 1, p. 81, 2021.

[2] A. Liyih, S. Anagaw, M. Yibeyin, and Y. Tehone, "Sentiment analysis of the Hamas-Israel war on YouTube comments using deep learning," *Sci. Rep.*, vol. 14, no. 1, p. 13647, Jun. 2024, doi: 10.1038/s41598-024-63367-3.

[3] "Understanding Public Opinions of Government Measures Against COVID-19 Through Twitter Sentiment Analysis," *Journal of Logistics, Informatics and Service Science*, vol. 11, no. 2, Feb. 2024, doi: 10.33168/JLISS.2024.0201.

[4] T. Arwindarti, E. I. Setiawan, and S. Imron, "Klasifikasi Sentimen Opini Publik Pada Instagram Pemerintah Kabupaten Bojonegoro Menggunakan LSTM," *Teknika*, vol. 13, no. 1, pp. 1–9, Nov. 2023, doi: 10.34148/teknika.v13i1.699.

[5] G. A. Sandag, E. H. E. Soegiarto, L. Laoh, A. Gunawan, and D. Sondakh, "Sentiment Analysis of Government Policy Regarding PPKM on Twitter Using LSTM," in *2022 4th International Conference on Cybernetics and Intelligent System (ICORIS)*, IEEE, Oct. 2022, pp. 1–6. doi: 10.1109/ICORIS56080.2022.10031474.

[6] V. R. Prasetyo, M. F. Naufal, and N. Soeratman, "Sentiment Analysis for Public Services Using Bidirectional Long Short-Term Memory Method," in *2025 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, IEEE, Jul. 2025, pp. 679–684. doi: 10.1109/ISITIA66279.2025.11137417.

[7] A. Radaideh and F. Dweiri, "Use of AI Applications to Learn the Sentiment Polarity of Public Perceptions: A Case Study of the COVID-19 Vaccinations in the UAE".

[8] I. Guellil *et al.*, "A Semi-supervised Approach for Sentiment Analysis of Arab(ic+izi) Messages: Application to the Algerian Dialect," *SN Comput. Sci.*, vol. 2, no. 2, p. 118, Apr. 2021, doi: 10.1007/s42979-021-00510-1.

[9] A. Younas, S. Riaz, S. Ali, R. Khan, M. Ullah, and D. Kwak, "Stacked LSTM model for contextual correlation detection among multiple emotions," *IEEE Access*, 2025.

[10] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artif. Intell. Rev.*, vol. 55, no. 7, pp. 5731–5780, Oct. 2022, doi: 10.1007/s10462-022-10144-1.

[11] R. A. Perdana, "Komparasi Algoritma Naive Bayes, Decision Tree, Dan K-Nearest Neighbor (K-Nn) Dalam Analisis Sentimen Cyberbullying Pada Komentar Instagram," Universitas Diponegoro, 2024.

[12] I. Guellil, F. Azouaou, and M. Mendoza, "Arabic sentiment analysis: studies, resources, and tools," *Soc. Netw. Anal. Min.*, vol. 9, no. 1, p. 56, 2019.

[13] K. Azim *et al.*, "Ensemble stacked model for enhanced identification of sentiments from IMDB reviews," *Sci. Rep.*, vol. 15, no. 1, p. 13405, Apr. 2025, doi: 10.1038/s41598-025-97561-8.

[14] L. Cao, "Sentiment Analysis of Social Media Text Based on Deep Learning," in *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, IEEE, Dec. 2023, pp. 1–5. doi: 10.1109/ICMNWC60182.2023.10435901.

[15] E. Novfuja, L. Efrizoni, E. Ali, and S. Susanti, "Prediksi Dukungan Publik Terhadap Program Makan Bergizi Gratis (MBG) Menggunakan Analisis Sentimen Berbasis Long Short-Term Memory (LSTM)," *Jurnal Algoritma*, vol. 22, no. 2, pp. 1704–1715, 2025.
[16] M. B. Musthafa, S. Huda, M. A. Ali, Y. Kodera, and Y. Nogami, "Evaluation of IDS model by improving accuracy and reducing overfitting using stacking LSTM," in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, 2024, pp. 1–5.
[17] C. S. K. Aditya, G. W. Wicaksono, and H. A. S. Heryawan, "Sentiment analysis of the 2024 presidential candidates using smote and long short term memory," *Jurnal Informatika Universitas Pamulang*, vol. 8, no. 2, pp. 279–286, 2023.
[18] A. Pratama and M. Rosyda, "Analisis Sentimen Dalam Aplikasi X Terhadap Pengungsi Rohingya Dengan Lstm," *SKANIKA: Sistem Komputer dan Teknik Informatika*, vol. 8, no. 1, pp. 95–105, Jan. 2025, doi: 10.36080/skanika.v8i1.3329.
[19] P. R. Panggabean, A. Asrianda, and H. A.-K. Aidilof, "Sentiment Analysis of Public Comments on X Social Media Related to Israeli Product Boycotts Using The Long Short-Term Memory (LSTM) Method," *Journal of Applied Informatics and Computing*, vol. 9, no. 3, pp. 775–783, 2025.
[20] N. C. Dang, M. N. Moreno-García, and F. De la Prieta, "Sentiment analysis based on deep learning: A comparative study," *Electronics (Basel).*, vol. 9, no. 3, p. 483, 2020.
[21] R. Olusegun, T. Oladunni, H. Audu, Y. Houkpati, and S. Bengesi, "Text Mining and Emotion Classification on Monkeypox Twitter Dataset: A Deep Learning-Natural Language Processing (NLP) Approach," *IEEE Access*, vol. 11, pp. 49882–49894, 2023, doi: 10.1109/ACCESS.2023.3277868.
[22] R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja, "The Impact of Features Extraction on the Sentiment Analysis," *Procedia Comput. Sci.*, vol. 152, pp. 341–348, 2019, doi: 10.1016/j.procs.2019.05.008.
[23] S. E. Alqaan and A. M. Qamar, "Sentiment Analysis of Arabic Tweets on Online Learning During the COVID-19 Pandemic: A Machine Learning and LSTM Approach," *Ingénierie des systèmes d information*, vol. 28, no. 6, Dec. 2023, doi: 10.18280/isi.280601.
[24] R. I. Kurnia and A. S. Girsang, "Classification of user comment using word2vec and deep learning," *Int J Emerg Technol Adv Eng*, vol. 11, no. 5, pp. 1–8, 2021.
[25] P. Dubey, N. Rakesh, P. Thakre, S. Matte, D. Tonde, and K. Dhawale, "Beyond Traditional LSTMs: A Deep Learning Model for Improved Sentiment Analysis," in *2025 12th International Conference on Emerging Trends in Engineering & Technology-Signal and Information Processing (ICETET-SIP)*, IEEE, 2025, pp. 1–6.
[26] A. Yadav and D. K. Vishwakarma, "Sentiment analysis using deep learning architectures: a review," *Artif. Intell. Rev.*, vol. 53, no. 6, pp. 4335–4385, Aug. 2020, doi: 10.1007/s10462-019-09794-5.
[27] W. Lin, H. Yu, and L. Wang, "A data-driven deep learning approach incorporating investor sentiment and government interventions to predict post-crash stock return in China's A-share market," *Journal of Innovation & Knowledge*, vol. 10, no. 3, p. 100704, May 2025, doi: 10.1016/j.jik.2025.100704.
[28] S. Najwa, S. Winarni, A. A. Pravitasari, and Y. Hidayat, "Emotion classification in social media posts related to telecommunication services using bidirectional LSTM," *Commun. Math. Biol. Neurosci.*, vol. 2025, p. Article-ID, 2025.
[29] M. P. Firdaus and D. Trisnawarman, "Analisis Sentimen Publik terhadap Program Tabungan Perumahan Rakyat Menggunakan Model IndoBERT Lite pada Komentar YouTube," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 5, no. 1, pp. 359–368, Jan. 2025, doi: 10.57152/malcom.v5i1.1744.
[30] M. Mujahid *et al.*, "Sentiment Analysis and Topic Modeling on Tweets about Online Education during COVID-19," *Applied Sciences*, vol. 11, no. 18, p. 8438, Sep. 2021, doi: 10.3390/app11188438.