SENTIMENT ANALYSIS OF E-COMMERCE TOKOPEDIA REVIEWS USING MT5

[1]Berlian Ishma Zhafira Sujana, [2]Ir. Indriati, S.T, M.Kom., [3]Rizal Setya Perdana, S.Kom., M.Kom., Ph.D.

[1,2,3] *Informatics Engineering Study Program, Faculty of Computer Science, Brawijaya University*
[1,2,3]*Jl. Veteran No.10-11, Ketawanggede, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145, Indonesia*

[1]berlianishma@ub.ac.id
[2]indriati.tif@ub.ac.id
[3]rizalespe@ub.ac.id

Abstract— **Sentiment analysis of Indonesian e-commerce reviews plays an important role in understanding user perceptions of products and services. Transformer-based models, particularly the Multilingual Text-to-Text Transfer Transformer (mT5), provide an effective text-to-text approach for sentiment classification tasks. However, model performance is highly influenced by the selection of training hyperparameters. Therefore, this study aims to analyze the effect of batch size variation and number of training epochs on the performance of the mT5 model in Tokopedia review sentiment classification. The dataset consists of Indonesian reviews categorized into two sentiment classes, positive and negative. The research stages include preprocessing, SentencePiece tokenization, mT5 model training, and evaluation using accuracy, precision, recall, and F1-score metrics. The experimental results show that a smaller batch size tends to produce more stable performance but requires longer training time, while a larger batch size requires more epochs to reach optimal convergence. The configuration using batch size 4 and 10 epochs achieves the best performance with a training accuracy of 0.963. These findings highlight the importance of proper hyperparameter selection in optimizing Transformer-based models for Indonesian sentiment analysis.**

Keywords— **sentiment analysis, mT5, transformer, hyperparameter, e-commerce.**

I. INTRODUCTION

The development of digital technology has driven a shift in Indonesian consumer behavior from conventional transactions to e-commerce platforms. By 2024, 65.6 million Indonesians were recorded as active e-commerce users, with Tokopedia as one of the largest platforms, with 158.35 million visits in the second quarter of 2022 [10]. This growth makes customer reviews a crucial factor in the consumer decision-making process. As many as 84% of consumers trust online reviews as much as recommendations from close friends [1], thus reviews play a strategic role in shaping perceptions of products and services.

Natural Language Processing (NLP)-based sentiment analysis offers a solution for managing large review volumes by classifying opinions into positive or negative categories. However, e-commerce reviews often contain mixed language, slang, technical terms, and foreign language borrowings, increasing the complexity of text processing. Conventional Large Language Model (LLM)-based approaches still face limitations in handling dynamic linguistic variations.

The Multilingual Text-to-Text Transfer Transformer (mT5) model offers a more adaptive approach because it was trained on the mC4 multilingual corpus, which includes 101 languages, including Indonesian [14]. Several studies have demonstrated the superior performance of mT5 in various NLP tasks, such as Spanish sentiment analysis with a Macro F1-score of 0.9781 [12], Indonesian Opinion Term Extraction with an F1-score of 88.09% [11], and multilingual propaganda detection with an accuracy of 99.61% [8].

Based on these findings, this study applies the mT5 model to Indonesian-language Tokopedia reviews for sentiment classification. The research focuses on analyzing the influence of hyperparameter variations, specifically batch size and number of epochs, on model performance to obtain an optimal configuration in the context of Indonesian-language digital reviews.

II. LITERATURE REFERENCES

A. Sentiment Analysis

Sentiment analysis is a branch of Natural Language Processing that aims to identify opinions or emotions contained in text and then classify them into positive or negative categories [6]. In the context of e-commerce, sentiment analysis plays a crucial role because customer reviews can be used as a basis for evaluating product quality and designing more targeted marketing strategies.

In this study, a sentence-level sentiment analysis approach was used, because e-commerce reviews generally consist of one or two short sentences.

B. Review

A review is a form of opinion written by a consumer about a particular product, service, or experience. In a digital context, reviews are typically published on e-commerce platforms, social media, and service provider websites. Reviews play a crucial role not only as a means of communication between consumers and service providers but also as a source of information for other potential buyers in making decisions [9].

C. Machine Learning

Machine learning (ML) is a branch of artificial intelligence that enables computer systems to learn patterns from data to generate predictions. In sentiment analysis, ML is used to

model the relationship between review text and sentiment labels, such as positive and negative. According to Mitchell (1997) [5], a system is said to be learning if its performance improves with experience, which in this context is represented by labeled review data.

The approach used includes supervised learning, as the training process is conducted using a dataset that has been labeled with sentiment. Through this process, the model learns linguistic patterns, emotional expressions, and context within the text with the goal of developing good generalization capabilities to new data.

D. Deep Learning

Deep Learning is a branch of ML that uses Artificial Neural Networks (ANN) with deep architecture. Deep Learning is capable of automatically learning highly complex data representations through feature learning [2]. Before the introduction of Transformer, sequence-to-sequence models were generally based on Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) with encoder-decoder architecture [13].

However, RNN models have limitations in processing long text contexts and struggle with parallelization. This issue was addressed with the emergence of the Transformer architecture, which replaces the sequential structure of RNNs with a self-attention mechanism to comprehensively understand the relationships between words.

E. Transformer

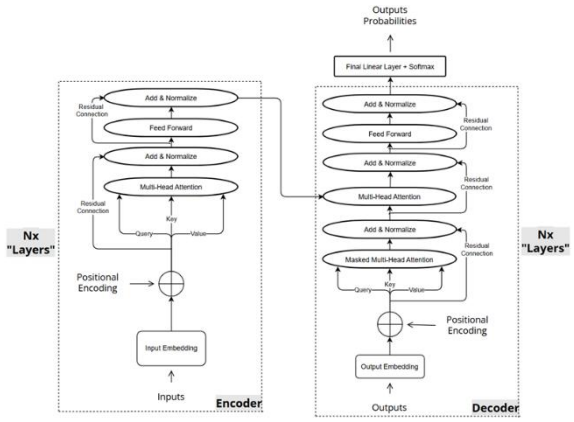

Fig. 1 Transformer Architecture [13]

A major turning point in the development of NLP occurred when Vaswani et al. (2017) introduced the Transformer — an attention-based neural network architecture. The Transformer replaces the need for sequential processing (as in RNN/LSTM) with a self-attention mechanism that allows the model to understand the relationships between words in a sentence in parallel. This architecture became the foundation for modern large language models such as T5 and mT5.

As shown in Fig. 1, the Transformer model has two main components: an Encoder on the left, which understands the context of the input text, and a Decoder on the right, which generates output text based on the encoder's representation. The advantages of this architecture lie in its ability to process long contexts, high parallelism, and large parameter scalability.

Input Embedding Layer: This layer transforms input in the form of words or subwords into a numeric vector representation. This representation is used to capture the semantic meaning of each token in the input text, allowing the model to understand the linguistic context of each element in the word sequence.

Positional Encoding: Transformer models, including T5 and mT5, lack the sequential nature of RNNs. Therefore, positional encoding is required to provide information about the relative position of tokens within a sentence. This process is achieved by adding positional values to the embedding results, allowing the model to distinguish between word sequences. This positional transformation typically uses sine and cosine functions to generate unique patterns at each token position. This allows the model to recognize the order and relationships between words within a sentence. Each encoder block in the Transformer architecture includes positional encoding after the embedding stage to maintain word order throughout the representation process.

Attention Mechanism: The Transformer's attention mechanism allows the model to contextually learn the relationships between words in a sentence. Through self-attention, each token can pay attention to other tokens in the same sequence, allowing the model to understand the semantic relationships between words.

For example, in a sentence containing the words "health" and "care," the model can recognize that they are related in meaning. In this mechanism, each token is represented by three vectors: query (Q), key (K), and value (V). The query describes the token being processed, the key represents all tokens in the sequence, and the value is used to calculate the final representation based on the relationship between the query and the key. If the query and key have a high match, then the attention score is also high, indicating that the token has strong relevance.

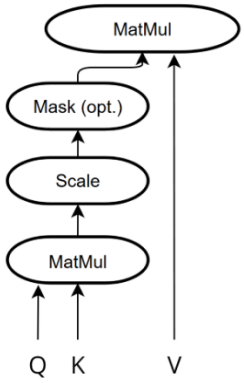

Fig. 2 Scaled Dot-Product Attention [13]

The attention mechanism (Fig. 2) calculates relationships between tokens using query, key, and value vectors. The process begins by multiplying the query and key to obtain a correspondence matrix that shows the extent to which each token is related to other tokens.

This multiplication result is then divided by the square root of the key dimension (d_k) to prevent the gradient values from becoming too large when passing through the softmax function. The softmax function is used to transform the result into a probability distribution, ensuring that the total attention weights sum to one.

This probability distribution is then used to weight the value vector, resulting in a final representation that highlights the words most relevant to the context. Mathematically, this mechanism can be expressed by equation 1.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_K}})V \quad (1)$$

The multi-headed attention mechanism is an extension of scaled dot-product attention that allows the model to process information from multiple representations in parallel. Instead of having a single attention function, the Transformer model divides it into multiple heads, each of which can learn to highlight different relationships between tokens.

Each head computes attention separately using different projection parameters for the query, key, and value. The results from all heads are then concatenate and projected back using the W_O weight matrix to produce a richer and more contextual final representation. Mathematically, this mechanism is expressed by equation 2:

$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W_o$(2)

with each head counted as:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3)$$

Feed-Forward Network: A Feed-Forward Network is used in each encoder and decoder block in the Transformer architecture. This component serves to enrich the representation of the results of the attention mechanism by performing non-linear transformations. In general, FFN consists of two linear layers with a ReLU activation function in between, and can be expressed by formula 4:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (4)$$

In the first stage, the input vector x is multiplied by the weight matrix W_1 and added with a bias b_1. The result is then passed to the ReLU activation function to add non-linearity. Next, the output of the ReLU function is multiplied by a second weight matrix W_2 and added with a bias b_2 to produce the final representation. This FFN layer helps the model in capturing complex relationships between features and expands the representational capacity of the network without relying solely on the attention mechanism.

Add and Normalization: These stages are the main components of each sub-layer of the Transformer architecture, both in the encoder and decoder sections. This process consists of two steps: addition and normalization (layer normalization).

First, the output from a sublayer (e.g., a multi-head attention or feed-forward network) is added to the initial input of that layer. This step ensures that important information from the input is preserved during the learning process.

Next, a normalization layer is applied to normalize the distribution of activation values in each layer. This normalization helps maintain network stability during training, accelerates convergence, and reduces the risk of unstable gradients. With the normalization layer, the model can maintain stability between layers despite numerous learning iterations.

This process is an essential part in maintaining a balance between the new information resulting from the transformation and the original information from the input, so that each Transformer layer remains efficient in learning complex data representations [13].

Residual Connection: Residual connection (skip connection) is a technique that directly connects the initial input of a sublayer to its output through a summation operation. The main purpose of this mechanism is to prevent the loss of important information from the input and to help maintain a stable gradient flow during the model training process.

In this way, the network can learn the changes (residuals) from existing representations, rather than having to build new representations from scratch. This approach is effective in addressing the vanishing gradient problem that often occurs in deep neural network models.

In Transformer, residual connections are applied after each multi-head attention and feed-forward network, before the normalization layer. The combination of residual connections and normalization layers helps the model maintain learning stability and retain information from previous layers.

Output Layer (Linear and Softmax): The final stage in the Transformer architecture is the linear output layer, which projects the vector representation from the decoder onto the vocabulary size dimension of the model [7]. Each element in the output vector represents a score (logit) for each token in the vocabulary.

The scores are then processed using the Softmax activation function to generate a probability distribution, resulting in a total probability of one. The token with the highest probability is selected as the prediction at each decoding step. In sequence generation models like mT5, this process is performed iteratively until a complete sequence of tokens is formed [13].

F. T5: Text-to-Text Transfer Transformer

T5 (Text-to-Text Transfer Transformer) is a large-scale language model developed by Google Research using a unified text-to-text framework approach, where all NLP tasks are formulated as processes that transform input text into output text [7]. This approach allows tasks such as translation, summarization, question-and-answer processing, and sentiment analysis to be completed within the same framework.

T5 is built using a Transformer architecture based on a self-attention mechanism that is capable of effectively capturing contextual relationships and long-range dependencies between tokens [13]. This model is trained on a large-scale corpus, giving it robust and flexible language representation capabilities for various NLP applications.

G. mT5: Multilingual Text-to-Text Transfer Transformer

mT5 is a multilingual version of T5 trained on the mC4 corpus, over 750 GB in size, covering 101 languages, including Indonesian [14]. This model maintains a text-to-text approach, allowing various NLP tasks to be formulated as text generation processes, including sentiment classification.

Architecturally, mT5 is based on the Transformer encoder–decoder [13]. The encoder generates a contextual representation of the input text using a self-attention mechanism, while the decoder generates output autoregressively by utilizing masked self-attention and cross-attention to the encoder representation. Compared to the classic Transformer, mT5 adopts relative position bias and pre-layer normalization to improve training stability [7].

mT5 supports fine-tuning to adapt pre-trained models to specific tasks using smaller, more targeted datasets. This study used the google/mt5-small variant (±300 million parameters) as the baseline model. This variant was chosen because it is more

computationally efficient than larger variants, thus aligning with the limitations of GPU resources and the size of the research dataset. Despite its lighter size, mT5-small still maintains sufficient multilingual capabilities to process Indonesian-language reviews that potentially contain language variations.

In mT5, text is processed using the SentencePiece tokenizer, which operates at the subword level without relying on specific language rules [3]. Each text is converted into a token, converted to a numeric ID, and then adjusted for length through padding and truncation. The model also uses end-of-sequence tokens as text boundary markers.

In the text-to-text approach, tasks are conditioned using instruction prefixes. In this study, prefixes such as "sentiment:" were used to direct the model to perform sentiment classification on review text.

H. Evaluation Metrics

Classification model evaluation was performed using confusion matrix-based metrics, consisting of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). This study used accuracy, precision, recall, and F1-score metrics [4].

Accuracy measures the proportion of correct predictions against all test data:

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

Precision shows the proportion of correct positive predictions:

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

Recall (sensitivity) measures the model's ability to detect all positive data:

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

F1-score is the harmonic mean between precision and recall:

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

Accuracy provides a general overview of performance, while precision and recall are more informative in imbalanced class distributions. The F1-score is used to assess the balance between a model's ability to identify positive classes and minimize prediction errors.

III. Methodology

A. Research Type

This research falls into the non-implementation category with an analytical/explanatory approach. Non-implementation research does not focus on building a new system, but rather on analyzing and evaluating an existing model, namely mT5. The analytical approach is used to examine the effect of variations in batch size and number of epochs on model performance in sentiment classification. This study aims to explain the relationship between training configuration and model performance based on the evaluation metrics used, thus obtaining an optimal configuration.

B. Research Methods

The strategy used was an experiment with the creation of NLP artifacts. Researchers collected a consumer review dataset, performed text preprocessing, applied mT5 model fine-tuning, and evaluated the sentiment classification results. The research design consisted of several main stages:

1. Collection of Indonesian language Tokopedia review dataset.
2. Data pre-processing to prepare text input.
3. Fine-tuning the mT5 model with the processed dataset.
4. Model evaluation with quantitative metrics.

C. Supporting Equipment

This research was conducted using software including Python 3.10 as the primary programming language. For natural language processing needs, this study utilized the Hugging Face Transformers and Tokenizers libraries. Additionally, the PyTorch and Scikit-learn libraries were used for machine learning implementation, as well as Pandas and NumPy for data processing. The model training and execution processes were run on Google Colab with GPU runtime support for optimal computing. Meanwhile, data visualization was performed using the Matplotlib and Seaborn libraries.

D. Data Collection

Data collection in this study did not involve a scraping process, but instead utilized a public dataset available on the Kaggle platform titled "PRDECT-ID: Indonesian Emotion Classification". This dataset contains consumer reviews written in Indonesian which served as the basis for sentiment analysis. The dataset contains 11 columns: Category, Product Name, Location, Price, Overall Rating, Number Sold, Total Reviews, Customer Rating, Customer Review, Sentiment, and Emotion. For this study, the two main columns used are "Customer Review", containing the review text written by consumers. Then the "Sentiment" column, containing positive and negative values.

E. Dataset Preprocessing

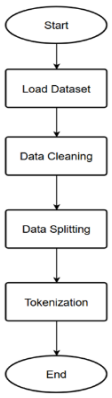

Fig. 3 Data preprocessing stage

The initial stage, as shown in the flowchart in Fig. 3, involves uploading the dataset to the Google Colab virtual environment. Next, data cleaning involves removing duplicates, rows with missing review text or null values, excess punctuation, and emojis.

After the cleaning process is complete, the data is then divided into three subsets, namely train, validation, and test with a ratio of 80:10:10. The data is divided randomly but still pays attention to label stratification to maintain class balance.

The final stage in preprocessing is text tokenization using the mT5 model's tokenizer. Each text review is converted into a sequence of numeric tokens using the tokenizer function, while the target labels are also tokenized to match the input-target format in seq2seq modeling.

F. Implementation of mT5

The training process was performed using Hugging Face's Trainer API. The model was initialized with the hyperparameter configuration shown in Table 1.

TABLE I
HYPERPARAMETER TRAINING MODEL MT5

Hyperparameter	**configuration**
learning_rate	5e-5
Epoch	3,5,10
Batch	4,8,16
max_input_length	128
max_target_length	8
weight_decay	0.01
predict_with_generate	True
eval_strategy	"epoch"
logging_steps	250
save_strategy	"epoch"
load_best_model_at_end	True
metric_for_best_model	“f1_macro”
generation_max_length	15

At each epoch, the model performs a forward pass to generate predictions, then calculates a loss value by comparing the predictions with the original labels. The loss value is used in the backward pass to update the model weights. After each epoch, an evaluation is performed using validation data, with the F1-macro metric as the primary reference. The best model is automatically saved through checkpointing and reloaded at the end of training using the load_best_model_at_end parameter.

The testing phase is conducted after training is complete using the trainer.predict() function on the testing data. The predicted results are compared with the original labels to calculate evaluation metrics such as accuracy, precision, recall, and F1-score. This evaluation provides a comprehensive overview of the model's final performance in classifying e-commerce review sentiment.

Overall, the implementation of mT5 in this study includes the model initialization stage, fine-tuning process, selection of the best model based on F1-macro, and final evaluation using testing data to measure the model's generalization ability.

G. Model Evaluation

After the training process was complete, the model was evaluated to determine the extent to which mT5-small was able to classify sentiment in Tokopedia user reviews. This evaluation was conducted using a testing dataset that the model had never seen during the training or validation process. The goal of this stage was to measure the model's generalization ability to new data.

This research focuses on sentiment classification, so the metrics used are classification evaluation metrics. The evaluation was conducted using a test set and calculated using several common classification performance metrics, namely Accuracy, Precision, Recall, and F1-Score. These metrics provide an overview of the model's accuracy in predicting "Positive" or "Negative" sentiment based on user reviews on the Tokopedia platform.

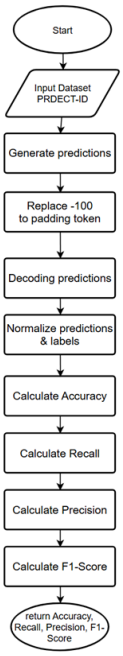

Fig. 4 Flowchart of mT5 model evaluation

All these metrics are calculated after the model generates predictions using HuggingFace Trainer's trainer.predict() function. The evaluation process follows the Evaluation flowchart (Fig. 4), which includes loading the test dataset, generating predictions, decoding the mT5 output, normalizing labels, and calculating all performance metrics.

IV. RESULT

TABLE III
HYPERPARAMETER TRAINING MODEL MT5

Batch Size	**epoch**	**Time (minute)**	**accuracy**		**F1-score**		**precision**		**recall**	
			Training	**validation**	**Neg**	**Pos**	**Neg**	**Pos**	**Neg**	**Pos**
4	3	14	0.904	0.902	0.903	0.904	0.949	0.863	0.862	0.950

	5	24	0.941	0.939	0.941	0.940	0.973	0.909	0.911	0.973
	10	57	0.963	0.957	0.964	0.962	0.978	0.947	0.950	0.977
8	3	22	0.874	0.846	0.878	0.870	0.888	0.860	0.869	0.880
	5	21	0.926	0.907	0.927	0.925	0.958	0.895	0.897	0.957
	10	26	0.944	0.955	0.945	0.944	0.977	0.913	0.915	0.977
10	3	13	0.859	0.868	0.852	0.866	0.948	0.794	0.773	0.953
	5	21	0.824	0.831	0.808	0.838	0.939	0.749	0.709	0.950
	10	47	0.957	0.944	0.958	0.956	0.978	0.937	0.940	0.977

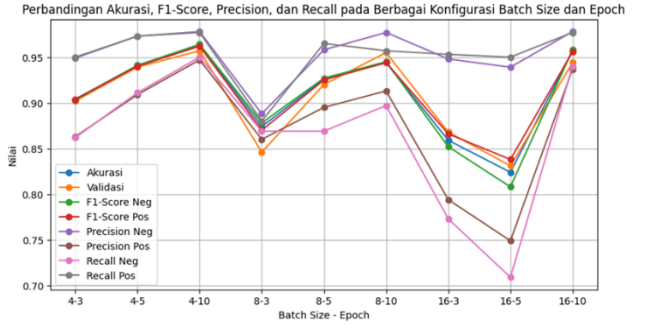

Fig. 5 Comparison of Accuracy, F1-Score, Precision, and Recall on Various Batch Size and Epoch Configurations

In Fig. 5, variations in batch size and number of epochs affect model performance, both on training and validation data. In general, increasing the number of epochs tends to improve accuracy, F1-score, precision, and recall.

At batch size 4, the model showed stable and consistent performance as the number of epochs increased. Training accuracy increased from 0.904 at 3 epochs, then 0.941 at 5 epochs, and finally to 0.963 at 10 epochs, while validation accuracy also increased from 0.902, 0.939, and finally to 0.957. The F1-score, precision, and recall values for both sentiment classes experienced a balanced increase, indicating that the model was able to learn sentiment patterns well. However, training time increased significantly, reaching 57 minutes at 10 epochs, indicating higher computational costs at small batch sizes.

At batch size 8, the model demonstrated a better balance between performance and time efficiency. At 10 epochs, the model achieved a high validation accuracy of 0.955 with relatively more efficient training time compared to batch size 4. Precision and recall values for the positive class tended to be high and stable, demonstrating the model's ability to consistently identify positive sentiments. Furthermore, the difference between training and validation accuracy was relatively small, indicating a lower risk of overfitting in this configuration.

Meanwhile, at a batch size of 16, model performance fluctuates at lower epoch counts. At 3 and 5 epochs, the accuracy and F1-score are relatively lower, indicating that the model has not yet fully converged. However, at 10 epochs, performance improves significantly, with a training accuracy of 0.957 and a validation accuracy of 0.944. However, the training time increases to 47 minutes, indicating a trade-off between improved performance and computational efficiency.

The test results show that hyperparameter selection plays a significant role in improving the performance of the mT5 model on Indonesian sentiment analysis tasks. Batch sizes that are too small tend to produce stable performance but require longer training times, while batch sizes that are too large require a larger number of epochs to achieve optimal convergence. Furthermore, increasing the number of epochs generally improves model performance, but the effect depends on the batch size used. These findings suggest that mT5 performance improvement is not solely determined by the training duration, but also by the balance between batch size and epochs used.

Based on the test results in Table II, the configuration with a batch size of 4 and 10 epochs produced the highest overall performance, with a training accuracy of 0.963 and high and balanced F1-score, precision, and recall values across both sentiment classes. This indicates that using a small batch size allows the mT5 model to learn sentiment patterns more deeply through more frequent parameter updates. This indicates good generalization capability, where the model is able to maintain performance on previously unseen data without any significant indication of overfitting.

V. Conclusions

Based on the implementation and evaluation results, the mT5-small model can be effectively applied to the task of sentiment analysis of Tokopedia e-commerce app reviews in Indonesian using a text-to-text approach. The implementation process includes preprocessing, tokenization, training using training data, F1-macro-based validation, and final evaluation using test data. The model is able to handle diverse review characteristics, including formal, informal, slang, technical terms, and a mixture of Indonesian and English.

Overall, this study proves that the mT5 model is effective in representing customer sentiment towards the Tokopedia e-commerce application, and shows that selecting the right hyperparameter configuration plays an important role in improving model performance on Indonesian language sentiment analysis tasks.

References

Bigcommerce.com, 2021. The Inside Scoop on Ecommerce Reviews: Why They Matter and How to Make the Most of Them. [online] Available at: https://www.bigcommerce.com/blog/online-reviews/ [Accessed 29 Aug. 2025].

Goodfellow, I., Courville, A. and Bengio, Y. (2016) Deep learning. Cambridge, Massachusetts: The MIT Press (Adaptive computing and machine learning).

Kudo, T. and Richardson, J. (2018) “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing,” in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Brussels, Belgium: Association for Computational Linguistics, p. 66–71. Available at: https://doi.org/10.18653/v1/D18-2012.

Manning, C.D., Raghavan, P. and Schütze, H. (2008) Introduction to information retrieval. Cambridge: Cambridge university press.

Mitchell, T. M. (1997) Machine learning. Nachdr. New York: McGraw-Hill (McGraw-Hill series in Computer Science).

Pang, B. and Lee, L. (2008) "Opinion Mining and Sentiment Analysis," Foundations and Trends® in Information Retrieval, 2(1–2), pp. 1–135. Available at: https://doi.org/10.1561/1500000011.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020) "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." arXiv. Tersedia pada: https://doi.org/10.48550/ARXIV.1910.10683.

Ragab, MI, Mohamed, EH and Medhat, W. (2025) "Multilingual Propaganda Detection: Exploring Transformer-Based Models mBERT, XLM-RoBERTa, and mT5."

Reisch, L.A. and Zhao, M. (2017) "Behavioural economics, consumer behavior and consumer policy: state of the art," Behavioral Public Policy, 1(2), pp. 190–206. Available at: https://doi.org/10.1017/bpp.2017.1.

Satudata.kemendag.go.id, 2024. GMV e-commerce market in Indonesia period 2019-2023. [online] Available at: https://satudata.kemendag.go.id/ringkasan/produk/perdagangan-digital-e-commerCe-indonesia-periode-2023 [Accessed 29 Aug. 2025].

Suchrady, RZ and Purwarianti, A. (2023) "Indo LEGO-ABSA: A Multitask Generative Aspect Based Sentiment Analysis for Indonesian Language," in 2023 International Conference on Electrical Engineering and Informatics (ICEEI). Bandung, Indonesia: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ICEEI59426.2023.10346852 .

Toledano López, OG, Madera, J., González Diez, H.R. and Simón-Cuevas, A., 2022. Fine-tuning mT5-based Transformer via CMA-ES for Sentiment Analysis. In: Recommendation System, Sentiment Analysis and Covid Semaphore Prediction for Mexican Tourist Texts (Rest-Mex 2022), A Coruña, Spain, September 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017) "Attention Is All You Need." arXiv. Available at: https://doi.org/10.48550/ARXIV.1706.03762 .

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A. and Raffel, C. (2021) "mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer," in Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online: Association for Computational Linguistics, p. 483–498. Available at: https://doi.org/10.18653/v1/2021.naacl-main.41.