Indonesian Abstractive Text Summarization Using Stacked Embeddings and Transformer Decoder

Edi Winarko, *Member, IAENG,* Luis Tanoto, Muhammad Haidar Reza

Abstract— **Document summarization can be categorized into two categories: extractive and abstractive summarization. Research in abstractive summarization is more limited than that of extractive summarization, especially for Indonesian documents. Most existing studies in Indonesian abstractive summarization rely on a single embedding approach in their encoder. This study aims to develop an abstractive Indonesian document summarization model using stacked embedding as an encoder and a Transformer-based decoder. Stacked embeddings offer the advantage of capturing a more comprehensive range of linguistic features, enhancing the model's ability to generalize across different word forms and morphological variations. The stacked embedding combines Bidirectional Encoder Representation from Transformers (BERT), Byte Pair Embedding (BPE), Character Embedding (CE), and FastText (FT). We conduct experiments to find the effect of BERT layer selection and various stacked embedding as an encoder in the proposed summarization model. Using the Liputan6 dataset, the experimental results show that using all layers of BERT as an encoder gives the best performance for summarization. In addition, the stacked embedding of BERT, CE, and BPE gives the highest F1 score of 35.58 (ROUGE-1), 15.40 (ROUGE-2), and 32.80 (ROUGE-L) when trained with 50,000 data. In contrast, when trained with 75,000 data, the stacked embedding performance is below BERT embedding, which has an F1 score of 37.18 (ROUGE-1), 18.19 (ROUGE-2), and 34.28 (ROUGE-L). Our proposed model achieves performance close to state-of-the-art models despite using less than 40% of the training data in Liputan6 dataset.**

Index Terms—**Abstractive summarization, Liputan6 dataset, Pre-trained embedding, Stacked embedding, Transformer decoder**

I. Introduction

INFORMATION is integral to human life, as it provides knowledge and new experiences. In today's digital age, we can quickly access vast amounts of information, such as global news articles, at any time [1]. However, these articles' increasing volume and complexity often make them lengthy and challenging to understand, discouraging readers and potentially leading to misinformation. There is a need for concise text summaries that capture the main points in a shorter format [2]. Manual summarization is time-consuming and can vary in quality based on age and comprehension. Thus, automatic text summarization is essential to efficiently condense information while preserving its original meaning and avoiding redundancy [3]. Developing models to summarize text automatically is a challenging task. This difficulty arises because machines do not possess the same deep and subtle understanding of the content humans use when creating summaries [4].

Document summarization can be categorized into two types: extractive and abstractive. Extractive summarization involves selecting and combining key sentences or points from the original text without altering its structure. In contrast, abstractive summarization generates a new summary by rephrasing the original text and modifying its structure while preserving the core meaning [5]. This approach often results in more natural and comprehensible summaries, as it accounts for the relationships between words and sentences [6]. However, abstractive summarization is more complex, requiring a deeper understanding of the entire text.

While extractive summarization has been extensively studied across various domains, including news articles [7], hotel reviews [8], and gadget reviews [9], recent years have seen a growing interest in abstractive summarization within the field of Natural Language Processing (NLP).

A study by [10] made significant progress in text summarization using a Sequence-to-Sequence model with Recurrent Neural Networks (RNNs). This approach achieved high ROUGE scores on benchmark datasets such as CNN/Daily Mail, Giga Word, and DUC. However, RNNs faced limitations, particularly in managing long-range dependencies in text, which prompted researchers to explore alternative methods for enhancing summarization techniques.

One of the key advancements in abstractive models is the attention mechanism introduced by [11] to address challenges in machine translation. This mechanism allows the source input to influence word generation with assigned weights. Many researchers subsequently incorporated this mechanism into their encoder-decoder architectures, achieving better

Manuscript received July 8, 2024; revised February 23, 2025.

This work is supported in part by the Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, UGM, Schema B Research Grant No. 3328/UN1/FMIPA.1.3/KP/PT.01.03/2024.

Edi Winarko is a lecturer at the Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia (Corresponding author, phone: +62-813-2766-3322; e-mail: ewinarko@ugm.ac.id).

Luis Tanoto is an undergraduate student at the Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia (e-mail: luis.tanoto@mail.ugm.ac.id).

Muhammad Haidar Reza is a postgraduate student at the Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia (e-mail: mhaidarreza@mail.ugm.ac.id).

results and higher ROUGE scores, as demonstrated by [12], [13], [14], among others. However, the attention mechanism did not resolve the issue of parallelization, which led to increased computation time and cost during training. To address this, [15] introduced the Transformer architecture in 2017, incorporating multi-head self-attention in the encoder-decoder structure to enable parallelization. This architecture has since been widely adopted and modified to improve model performance, such as in the RC-Transformer by [5] and enhancements to traditional Sequence-to-Sequence models by [16].

Inspired by the Transformer's ability to capture feature representations, [17] introduced Bidirectional Encoder Representations from Transformers (BERT), initially as a pre-trained word embedding model for various NLP tasks. BERT outperformed 11 NLP tasks at the time [17], and experiments were conducted to examine the effect of BERT layer selection for the Named Entity Recognition task. Today, BERT has become both a word embedding technique and a standalone model in the NLP field, with text summarization being no exception. Reference [18] applied BERT as an encoder in the traditional Transformer architecture, achieving an average ROUGE-1, ROUGE-2, and ROUGE-L score of 33.48 on the CNN/DM dataset. Similarly, [19] introduced the BERTSUM architecture by modifying and fine-tuning BERT for summarization tasks, achieving a ROUGE-1 F1 score of 41.72 on the CNN/DM dataset and 48.92 on the New York Times dataset.

In addition to English-language datasets, many researchers have explored text summarization in non-English languages. For instance, [20] applied the pointer-generator approach to the LCSTS (Large-scale Chinese Short Text Summarization) dataset for Chinese text summarization. Similarly, [21] investigated summarization for Japanese texts by employing BERT as an encoder and a Transformer-based decoder to summarize Japanese documents.

Indonesia, the world's fourth most populous country, produces abundant information daily. There is also research on extractive summarization in Indonesian news [22] and web content [23]. However, current research in abstractive summarization for Indonesian documents is relatively limited. For instance, [24] used BiGRU as the main component in their encoder-decoder model and applied it to Indonesian journal texts with a performance of 12%. A study by [25] applied a weighted genetic algorithm and MSOP to Indonesian articles, but the results were insufficient. In 2020, [26] created a large-scale Indonesian dataset from Liputan6.com's news articles and proposed the BertAbs and BertExtAbs models. These models achieved F1 scores of 40.94 (ROUGE-1) and 41.08 (ROUGE-1), respectively. Similarly, [27] applied this approach to the IndoSum dataset [28].

Most abstractive summarization models utilize one word embedding in the encoder, such as Word2Vec, FastText, and BERT. While individual embedding has strengths, it also has limitations that can affect summarization quality. One approach used in several NLP tasks is utilizing stacked embeddings to replace individual word embeddings in the encoder. The use of stacked embeddings has yielded better performance in sequence tagging models (NER) [29] and aspect-based sentiment analysis models [30].

The main contribution of this paper includes:

1. We study the effect of BERT layers for abstractive summarization tasks. This study is motivated by the result of [17], which suggests that BERT layer selections affect the model performance in different tasks. Furthermore, we use the results of these experiments as our baseline.
2. We propose a novel abstractive summarization model that utilizes stacked embeddings as the encoder and a Transformer-based decoder. The encoder combines four different types of embeddings: BERT [17], Byte Pair Encoding (BPE) [31], Character Embedding (CE) [32], and FastText [33]. This combination aims to capture a comprehensive range of linguistic features, enhancing the model's ability to generate high-quality summaries.

The remainder of this paper is organized as follows. Section 2 discusses the research methodology applied to this study. Section 3 presents and discusses the experimental results, and Section 4 concludes this research paper.

II. Methodology

Fig. 1 is the workflow diagram that outlines the process of building and evaluating a text summarization model [34]. It begins with data collection, followed by data preprocessing steps such as removing irrelevant text and case folding. After preprocessing, stacked embeddings are created for training, validation, and test datasets. The training and validation data are used to train and fine-tune the model, resulting in a trained model. The model is then evaluated using the test data, and its performance is assessed using the ROUGE metric, which measures the quality of the generated summaries.

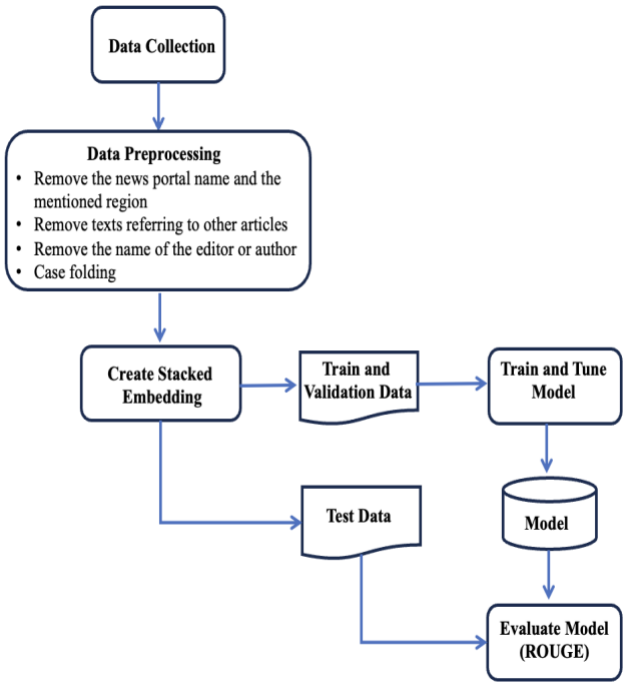

Fig. 1. The process of building and evaluating abstractive summarization models

A. Data Collection

The dataset used in this research, obtained from [26], consists of 215,827 news articles from Liputan6.com, covering the period from October 2000 to October 2010. Each article is stored in a JSON file with five key attributes:

`clean_article` (the main content of the news article), `clean_summary` (an abstractive summary of the article), `extractive_summary` (an extractive summary of the article), `id` (a unique identifier for the article), and `URL` (the link to the article's webpage). The articles and summaries are tokenized by splitting on whitespace and punctuation.

For processing efficiency, the dataset is converted to CSV format, retaining only the essential fields: `clean_article` and `clean_summary.` The original data has been split into training, validation, and test sets (see the second column of Table I). Due to computational resource constraints, only a subset of the training and validation data is used in this research. To create these subsets, we selected articles containing 300 tokens or fewer. Table I shows the allocation of research data across different datasets. For reduced datasets, the "50K Dataset" includes 50,000 training samples, 7,500 validation samples, and 10,972 test samples, while the "75K Dataset" contains 75,000 training samples, 7,500 validation samples, and 10,972 test samples. This allocation allows for consistent test sample size across different training dataset sizes for comparative analysis.

TABLE I
THE ALLOCATION OF RESEARCH DATA

Data	Original Data	50K Dataset	75K Dataset
Training	193,883	50,000	75,000
Validation	10,972	7,500	7,500
Test	10,972	10,972	10,972

TABLE II
NUMBER OF TOKEN STATISTICS FOR TRAIN, VALIDATION, AND TEST DATA

Statistics	Original Training	50K Training	75K Training	Validation	Test
Mean	184	152	152	164	172
Std	114	56	56	45	105
Min	8	8	8	34	6
Q1	114	110	110	132	124
Q2	153	142	142	158	151
Q3	219	187	187	190	187
Max	6717	300	300	300	2220

Table II shows the length of token statistics of articles in the original, 50K, and 75K training data, validation data, and test data. This table shows that the filtered training sets (50K and 75K) are more uniformly distributed than the original training data, with reduced variability and truncated maximum lengths. The original training data has a wider range and more variability in token counts. The identical statistics for 50K and 75K training sets indicate consistent sampling (filtering) is applied regardless of the sample size. The validation set contains longer sequences than the training sets, which could test the model's ability to generalize to slightly longer inputs. The test set has much longer sequences (max 2220) than training and validation data, which could assess how well the model handles sequences beyond its training range.

B. Data Preprocessing

A cleaning process is conducted before feeding the data into the feature extractor for numerical representation. The preprocessing involves removing unnecessary parts of the documents, such as the news portal name, mentioned regions typically found at the beginning or end, references to other articles (e.g., '[baca: …]' meaning 'read'), and the author's or editor's names usually at the start or end. This step ensures clarity and relevance. Punctuation is retained to preserve the document's meaning, and text is converted to lowercase to standardize representation, avoiding discrepancies between words like "*Makan*" and "*makan*" ('eat'). We do not remove stopwords, and no stemming or lemmatization is applied.

C. Stacked Embedding

Stacked embeddings combine different embeddings by concatenating them into a single vector. For example, if we stack a 768-dimensional BERT embedding with a 300-dimensional FastText embedding, we get a 1068-dimensional concatenated model. We explore four stacked embedding encoders consisting of BERT embedding, Byte Pair Embedding (BPE), Character Embedding (CE), and FastText (FT) embedding (see Table II). We use a BERT-based encoder as our baseline.

To create BERT embedding, we use the Indonesian BERT model '*indobenchmark/indobert-base-p1*', a state-of-the-art language model for Indonesian based on the BERT model [35]. The embedding dimension of this BERT model is 768. Byte-Pair Encoding (BPE) can effectively capture common and rare word patterns in text. BPE works by merging the most frequent pairs of characters into subwords, enabling the model to learn meaningful representations at different levels of detail. We use the BPE model called BPEmb, developed by [36]. BPEmb is a collection of pre-trained subword embeddings for 275 languages based on BPE and trained on Wikipedia. It offers embeddings of various sizes: 25, 50, 100, 200, and 300 dimensions. In our work, we use the 100-dimensional embeddings.

Learning character-level embeddings has the benefit of creating representations suitable for the specific NLP task and domain. Character embeddings (CE) are especially useful for languages with rich morphology and for handling out-of-vocabulary (OOV) words. The model architecture we use to generate word embeddings from characters is based on the model described in [30]. In this model, the character-based representation of words has a dimension of 50.

The FastText model, developed by Facebook's AI Research (FAIR) lab [33], is based on the Skip-gram model from Word2Vec. Its subword approach represents each word as a collection of character n-grams. Special symbols < and > are added at the beginning and end of words to distinguish prefixes and suffixes from other character sequences. The word itself is also included in its set of n-grams. Once all n-grams for a given word are extracted, a vector representation is assigned to each n-gram. The final word representation is then obtained by summing the vector representations of all the n-grams. In this study, we utilize pre-trained FastText embeddings (*cc.id.300.bin.gz*), trained on Common Crawl and Wikipedia, with a dimension of 300 [37]. To create and stack embeddings, we employ the FlairNLP framework [38], which facilitates the combination of embeddings from multiple sources through a simple API.

Table III shows the sizes of stacked embeddings for five models. Each model builds on the basic BERT model, which has a size of 768. The other models add different elements to BERT, making their sizes larger. Model 2 combines BERT

with BPE, resulting in a size of 868. Model 3 combines BERT with CE and BPE, resulting in a size of 918. Model 4 pairs BERT with FT, leading to a size of 1068. Model 5 combines BERT with FT and BPE, giving the largest size of 1168.

TABLE III
TYPES OF STACKED EMBEDDING AND THEIR EMBEDDING SIZE

Model	Stack Embedding	Size of Stacked Embedding
1	BERT	768
2	BERT+BPE	868 (768 + 100)
3	BERT+CE+BPE	918 (768 + 50 + 100)
4	BERT+FT	1068 (768 + 300)
5	BERT+FT+BPE	1168 (768 + 300 + 100)

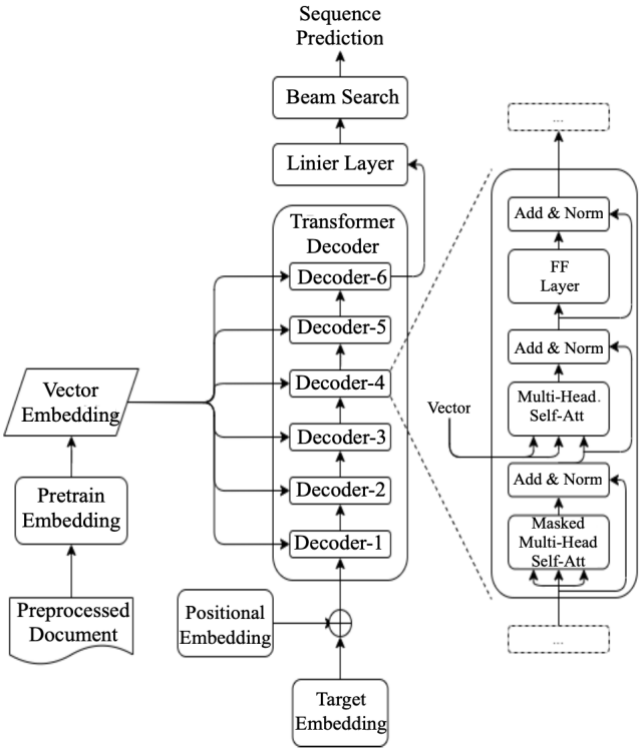

Fig. 2. Proposed model

D. Proposed Model Architecture

Fig. 2 shows the architecture of the proposed model. The decoder of the proposed model is similar to [21], [26], [27], while the encoder is the stack of pre-trained embeddings. The detailed view of each decoder is shown on the right side of the figure. Since we use pre-trained embedding as an encoder, we do not need to fine-tune the encoder. The output of stacked embedding is a vector representation of tokens for the particular document. Since the representations generated do not consider the maximum sequence length for the whole document, the representations are padded (or, for some documents, truncated) until reaching the length of 300 (for source articles) and 150 (for target summaries). Therefore, when entering the model architecture, each document will have the exact dimensions. For example, a stack embedding consisting of BERT and BPE embeddings provides a document representation of size equal to $1 \times 300 \times 868$ for the source articles and $1 \times 150 \times 868$ for the target summaries.

The padded vector representations are then passed through each decoder block to be calculated with the representation from the previous decoder block. The teacher-forcing method is employed during the training process. It enables input from the first block of the decoder to come from the right-shifted target summaries regardless of what word is predicted in the decoder at the corresponding position. The leftmost token from the right-shifted target summaries is replaced with [CLS] or [SOS] token as a flag for the start of the sequence.

Unlike the encoder, which utilizes different stacked embeddings for each scenario, the decoder only uses the BERT Tokenizer to tokenize the target summary to token chunks for all scenarios. The tokenized document is fed into embedding layers to get the vector representation, and the result is added to the representation from positional encoding to form the final vector representation for the first block decoder. As Transformer architecture removes the recurrent concept from it, the model cannot obtain a proper word order that is important in sequence tasks; hence, it is required to have this positional encoding to replace that functionality. Equations (1) and (2) show the positional encoding used in this research for odd and even indices.

$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (1)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (2)$$

In these equations, *pos* is the current position, i is the dimension index, and d is the model's dimension (embedding size).

Each decoder block receives two inputs: one from the encoder, which is the source sequence representation, and another from the previous decoder block. For the first decoder block, this second input is the target embeddings, and for the remaining blocks, it's the output from the previous decoder block. Within each block, the inputs are processed through different layers. The decoder input first goes to the masked multi-head self-attention layer, where it is split into three parts: Q (query), K (key), and V (value). These are used in the scaled-dot product attention along with a look-ahead mask, ensuring that the decoder only attends to tokens up to the current position in the sequence. After that, a residual connection and layer normalization (also called "Add & Norm") is applied. This masked attention allows the decoder to focus only on the tokens it has generated so far without looking at future tokens.

The masked attention formula is shown in Equation (3), where d_k is the vector of K's dimension, and the *Mask* is the mask vector used to mask out certain positions, for example, padding tokens or future tokens.

$$MaskedAttention(Q, K, V, Mask) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + Mask\right)V \quad (3)$$

The masked self-attention is performed using multiple heads, allowing the decoder to focus on different parts of the target sequence at once. Equations (4) and (5) show the formula of masked multi-head self-attention. W^O is a learnable matrix that projects the concatenated output back to

the model's original dimension. The Q_i, K_i, and V_i are each attention head's projected queries, keys, and values.

$$MHA(Q,K,V) = Concat(head_1, \dots, head_n)W^O \quad (4)$$

$$head_i = Attention(Q_i, K_i, V_i) \quad (5)$$

Next, the output from the first step is passed through the multi-head cross-attention layer. In this step, the queries (Q) come from the decoder output, while the keys (K) and values (V) come from the encoder output. These are used in scaled-dot product attention along with a padding mask, which ensures that padding tokens in the input are ignored. This allows the decoder to attend to the most relevant parts of the input sequence (encoder output) while generating the target sequence. After that, the results are recombined and passed through layer normalization (Add & Norm layer) to produce a new, refined representation.

Last, the result from the second calculation will pass through the position-wise feed-forward layer to get an additional calculation, followed by normalization. The final representation from the last layer is passed into the next decoder block. The final representation from the last decoder block will go through the linear layer. This layer enables the representation to be mapped into a logit vector, which contains probabilities of words in the dictionary in a particular position. Subsequently, the vector will go through the softmax layer to be normalized towards the overall probability distribution, and the highest one is selected as the word prediction for that position. The same process goes on until reaching the end of the sequence. Despite using the correct target word for each position to be passed into the decoder block, the word generation or prediction is still used to calculate the loss value.

Configurations of the decoder are similar to [21], [27], consisting of 6 stacked decoder layers, 12 layers of multi-head self-attention, 2,048 neurons for each hidden layer in the feed-forward model, drop out = 0.1, initial learning rate = 0.0001, and warm-up steps = 4000.

The model is trained for ten epochs, and the vocabulary size equals the number of vocabularies formed in the BERT Tokenizer. In this model, we use the cross-entropy loss and Adam optimizer and employ the learning rate scheduler used by [21]. Equation (6) shows what the learning rate (lr) scheduler looks like, where *init_lr* is the initial learning rate used, cs is the current step, and ws is the warm-up step.

$$lr = init\_lr \cdot \frac{min(cs^{-0.5}, ws^{-1.5} \cdot cs)}{ws^{-0.5}} \quad (6)$$

The model fine-tuning process involves a combination of varying the initial learning rate and adjusting the number of warm-up steps. The initial learning rate values are tested at 0.001, 0.0001 (default), and 0.00001, while the warm-up steps range from 1000, 4000 (default), to 10000. Each time the model's performance improves during the validation stage, the system saves the result as a checkpoint, which includes the updated model weights and optimizer states.

E. Model Evaluation

We evaluate the models using test data consisting of 10,972 rows. The evaluation process is pretty similar to the training phase. It starts by initializing the object to define the pre-trained embedding (including the stacked embeddings), padding the vector representation according to the encoder's maximum sequence length, and inferencing the result in the decoder. However, the treatment on the decoder side is quite different from the training since the input to the first decoder block is only a list containing the CLS/SOS token. Besides that, the beam search algorithm is applied at the top of the architecture right after mapping the final representation to the logit vector. This algorithm is also equipped with a trigram blocking mechanism to suppress the word repetition (a penalty is given to the word generated for more than three consecutive occurrences so that the word is blocked in the next occurrence). The beam width used in this study equals three and only keeps the best sequence at the end (n_best=1). The word prediction generated by the decoder and beam search algorithm for the corresponding position will be appended to the list mentioned above and become the decoder's input for the next position. This process will continue until the SEP/EOS token is predicted or the pre-defined maximum sequence length (150) is reached.

After obtaining the inference results, they are evaluated using the ROUGE (Recall Oriented Understudy for Gisting Evaluation) score. The ROUGE score measures the number of matching sequences (i.e., n-gram, LCS) between the generated and reference summaries [39], ranging between 0 and 1. In this paper, three types of ROUGE are being utilized, namely R1(ROUGE-1), R2 (ROUGE-2), and RL (ROUGE-L), as they are the commonly used ROUGE metrics in summarization tasks [16].

R1 measures the overlap of unigrams (individual words) between the generated summary and the reference summary, and R2 does the same for bigrams (pairs of consecutive words). In contrast, RL evaluates the similarity between the generated and reference summaries by identifying the longest common subsequence of words, which captures the sequence similarity between the texts. Unlike R1 and R2, which rely on counting word matches, RL uses a string-matching technique to assess the alignment of word sequences.

III. Results and Discussion

In this section, we first examine the impact of selecting different BERT layers on the model's performance, where BERT embeddings are utilized exclusively as the encoder. Next, we assess the effect of employing stacked embeddings as an encoder on model performance using training datasets of 50K and 75K samples. We then compare the proposed model with existing models to evaluate their relative effectiveness. Finally, we will analyze the results of the summarization.

A. The effect of BERT layers selection

The BERT model consists of 13 stacked layers. The first layer is an embedding layer, while the remaining 12 layers are bidirectional encoder layers. Information flows from the lower layers to the higher ones, with each layer refining the representation of the input until the final layer produces a numerical vector for each word. In a study by Devlin et al. [17], the impact of different BERT layers on the performance of the named entity recognition (NER) task was analyzed. In this paper, we adopt a similar approach to examine the effect of layer selection on the summarization task.

Table IV presents the accuracy and loss metrics for various

BERT layers during both training and validation after ten epochs. The results demonstrate that using the sum of all layers achieves the highest training accuracy (49.060) and the lowest training loss (3.175), alongside the best validation accuracy (36.369) and a relatively low validation loss (4.485). These findings suggest that aggregating information from all BERT layers provides the most comprehensive and effective task representation. In contrast, using individual layers or subsets of layers results in diminished performance, with the first layer showing the lowest accuracy and highest loss during both training and validation. The sum of the last four layers also performs well, indicating that deeper layers capture more relevant features for the summarization task. Based on these results, we adopt the sum of all layers when constructing BERT embeddings.

TABLE IV
ACCURACY AND LOSS OF BERT LAYERS DURING TRAINING AND VALIDATION

BERT layers	Training		Validation	
	Acc	Loss	Acc	Loss
First layer	34.323	4.084	25.127	5.256
Last layer	42.440	3.616	31.164	4.838
Second last layer	43.368	3.563	32.791	4.753
The last four layers	45.790	3.405	34.333	4.648
All layers	**49.060**	**3.175**	**36.369**	**4.485**

A. The Effect of Stacked Embedding Using 50K Train Data

Fig. 3 compares the performance of five stacked embedding encoders during training and validation. In all cases, the training loss consistently decreases across epochs, which indicates that the model is learning effectively from the training data. The validation loss also decreases, but not always at the same rate as the training loss. The gap between training and validation loss provides insight into the model's generalization capabilities. The BERT model shows the smallest loss difference, indicating a stable performance with minimal overfitting signs. Similarly, BERT+BPE has a comparable loss difference, which also reflects limited overfitting and good generalization ability. BERT+CE+ BPE**,** BERT+FT**,** and BERT+FT+BPE show signs of overfitting. Complementing this visual analysis, Table V provides a detailed comparison of the best epoch and the performance of the stacked embeddings encoder during both training and validation.

It can be seen in Table V that BERT achieves a reasonable balance between training and validation accuracy while also maintaining a relatively close match between training and validation loss. BERT+BPE slightly improves the accuracy of training and lowers the loss. BERT+CE+BPE provides a notable improvement in both training accuracy and validation accuracy. The BERT+FT model exhibits a high training accuracy with a more significant drop in validation accuracy and a more noticeable difference between training and validation loss, indicating that it overfits the most among the models. Based on the analysis result of the graph in Fig. 3 and Table V, the BERT+FT+BPE model shows the best overall performance in both the graph and the table. The graph confirms that it achieves the lowest validation loss and the smallest gap between training and validation, reflecting the best generalization. The BERT+CE+BPE model balances training and validation losses, confirming its solid performance, as seen in the table.

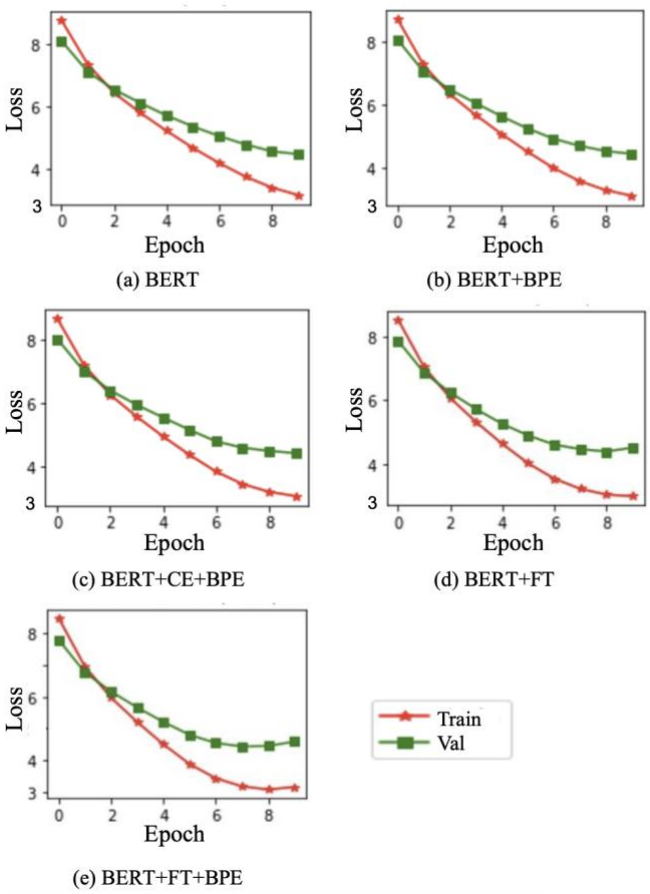

Fig. 3. Training and validation loss of five stacked embeddings encoder on 50K training data

TABLE V
THE ACCURACY AND LOSS OF STACKED EMBEDDINGS ENCODER DURING TRAINING AND VALIDATION (50K TRAIN DATA)

Stacked Embedding	Best Epoch	Training		Validation	
		Acc	Loss	Acc	Loss
BERT	10	49.060	3.175	36.369	4.485
BERT+BPE	10	49.282	3.125	36.399	4.442
BERT+CE+BPE	10	49.835	3.069	36.744	4.416
BERT+FT	9	49.306	3.084	36.232	4.453
BERT+FT+BPE	9	**49.941**	**3.058**	**36.952**	**4.402**

Table VI presents the ROUGE scores for various models on test data, focusing on Precision (P), Recall (R), and F1-score (F1) for R1, R2, and RL. The BERT+CE+BPE model appears to perform best across most metrics. It consistently achieves the highest precision (P) and F1 scores in R1, R2, and RL. The BERT+FT+BPE model shows strong performance in recall (R) scores, achieving the highest values for R1-R (36.96), R2-R (16.83), and RL-R (34.01). In most metrics, BERT+FT offers improvements over the base BERT model but does not reach the top performance levels of BERT+CE+BPE or BERT+FT+BPE. These results suggest that stacking embeddings, particularly combining BERT with CE and BPE, can lead to improved performance in text analysis tasks as measured by ROUGE scores.

TABLE VI
THE ROUGE SCORE OF STACKED EMBEDDINGS ENCODER ON TEST DATA WHEN THE MODELS TRAINED USING 50K TRAIN DATA

Metrics		BERT	BERT+ BPE	BERT+ CE+BPE	BERT +FT	BERT+ FT+BPE
R1	P	34.49	34.72	**35.35**	35.21	34.98
	R	36.94	36.40	36.86	36.35	**36.96**
	F1	35.18	35.02	**35.58**	35.26	35.41
R2	P	14.00	14.12	**14.64**	14.56	14.52
	R	16.64	16.38	16.78	16.48	**16.83**
	F1	14.98	14.92	**15.40**	15.21	15.34
RL	P	31.71	31.96	**32.60**	32.35	32.21
	R	33.94	33.49	33.98	33.37	**34.01**
	F1	32.33	32.22	**32.80**	32.38	32.59

TABLE VII
F1 SCORE OF TUNING BERT+CE+BPE ON THE COMBINATION OF LEARNING RATE AND WARM-UP STEP (50K TRAIN DATA)

Learning Rate	Warm-up Steps	R1	R2	RL
0.001	1000	5.62	0.1	5.6
	4000	18.16	2.49	17.04
	10000	25.97	6.55	23.76
0.0001	1000	32.35	10.88	29.68
	25000	35.23	15.10	32.47
	4000	**35.58**	**15.40**	**32.80**
	5000	35.36	15.30	32.55
	7500	34.52	13.90	31.81
	10000	33.56	12.49	30.72
0.00001	1000	28.83	7.07	26.17
	4000	27.34	5.92	24.97
	10000	22.48	3.45	20.74

We tried to finetune the BERT+CE+BPE model using different combinations of learning rates and warm-up steps. The F1 scores on the test data after finetuning are shown in Table VII. The optimal combination is found with a learning rate of 0.0001 and 4000 warm-up steps, achieving the highest F1 scores across all metrics (R1: 35.58, R2: 15.40, RL: 32.80). Increasing the learning rate to 0.001 results in significantly lower F1 scores, suggesting that a learning rate that is too high leads to inadequate convergence and poorer model performance. Conversely, decreasing the learning rate to 0.00001 also degrades performance, indicating that a learning rate that is too slow might not provide sufficient updates for effective learning within the given steps. Furthermore, the number of warm-up steps also critically impacts performance, with both too few (1000) and too many (10000) warm-up steps leading to suboptimal results. Thus, a moderate learning rate and an appropriately chosen number of warm-up steps are crucial for optimizing the model's performance. After conducting finetuning, we find that the optimal hyperparameters of the BERT+CE+PBE model are similar to the default settings used in Table VI.

B. The Effect of Stacked Embedding Using 75K Train Data

Fig. 4 shows the performance of five stacked embedding encoders on 75K training data, with both training and validation losses decreasing over the epochs, indicating consistent learning and improvement across all models. BERT+FT+BPE exhibits the smallest loss difference, indicating the best generalization and least overfitting among the models. BERT and BERT+CE+BPE also show reasonable generalization with moderate overfitting. In contrast, BERT+BPE and BERT+FT display the highest loss differences, indicating stronger overfitting, with BERT+FT being the most affected.

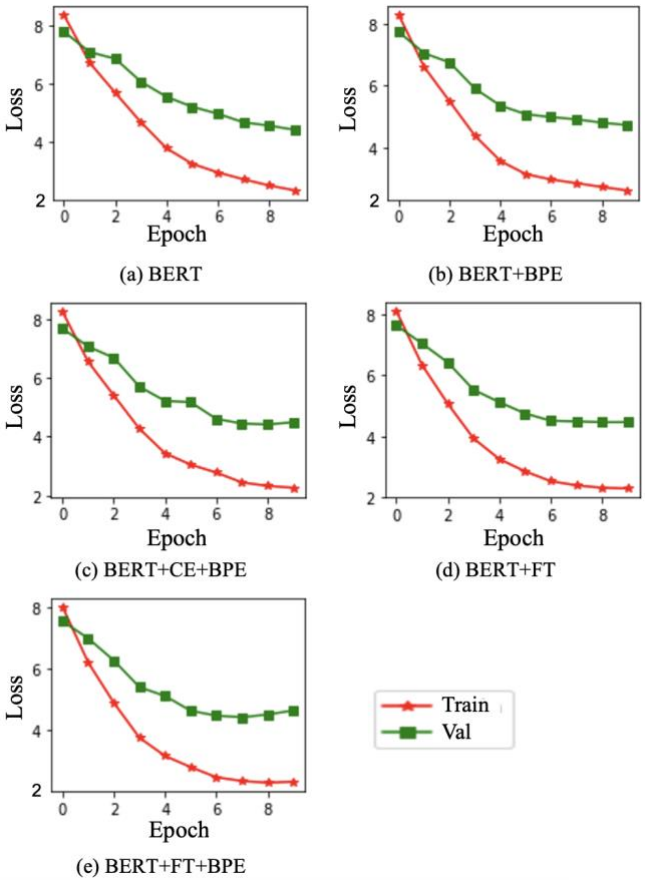

Fig. 4. Training and validation loss of five stacked embeddings on 75K training data

TABLE VIII
THE ACCURACY AND LOSS OF STACKED EMBEDDINGS ENCODER DURING TRAINING AND VALIDATION (75K TRAIN DATA)

Stacked Embedding	Best Epoch	Training		Validation	
		Acc	Loss	Acc	Loss
BERT	10	61.557	2.333	39.642	4.418
BERT+BPE	10	57.627	2.593	36.688	4.719
BERT+CE+BPE	9	**62.355**	2.316	**40.352**	**4.408**
BERT+FT	9	62.062	**2.312**	39.366	4.466
BERT+FT+BPE	8	61.754	2.347	39.833	4.428

Table VIII offers a detailed comparison of the best epoch and the performance of the stacked embedding encoders in both training and validation. BERT+CE+BPE is the best-performing model, achieving the highest validation accuracy and demonstrating strong generalization across the dataset, as shown in Fig. 4 and Table VIII. BERT+FT+BPE also performs well, achieving good validation accuracy and generalization, though it slightly lags behind BERT+CE+BPE. BERT+BPE performs worse than plain

BERT. These results highlight the benefits of using a mix of embeddings to improve the model's ability to understand and generate summaries, with BERT+CE+BPE standing out as the most effective combination for training and validation.

Table IX presents the performance of various stacked embeddings on test data. BERT+FT achieves the highest precision across all metrics (R1: 31.11, R2: 14.73, RL: 28.61), indicating its effectiveness in generating accurate summaries. However, the baseline BERT outperforms others in the recall, particularly for R1 (49.51) and RL (45.68), which suggests its strength in capturing a broader range of relevant content from the reference summaries. Regarding the F1-score, BERT+FT shows slightly superior performance in R1 (37.19) and R2 (18.11), while BERT and BERT+FT+BPE combinations exhibit competitive results. While BERT+FT stands out for precision and balanced performance, the original BERT model remains strong in recall, highlighting its comprehensive content capture capabilities.

TABLE IX
THE ROUGE SCORE OF STACKED EMBEDDINGS ON TEST DATA WHEN THE MODELS TRAINED USING 75K TRAIN DATA

Metrics		BERT	BERT +BPE	BERT+ CE+BPE	BERT +FT	BERT+ FT+BPE
R1	P	30.47	30.62	30.40	**31.11**	30.61
	R	**49.51**	48.10	48.90	48.07	48.42
	F1	37.18	36.78	36.95	**37.19**	36.96
R2	P	14.48	14.21	14.34	**14.73**	14.40
	R	**25.60**	24.41	25.08	24.63	24.77
	F1	**18.19**	17.62	17.95	18.11	17.90
RL	P	28.09	28.21	27.98	**28.61**	28.18
	R	**45.68**	44.36	45.05	44.26	44.63
	F1	**34.28**	33.90	34.02	34.21	34.04

Based on the fine-tuning results with 50K training data, the optimal configuration was determined to be a learning rate of 0.0001 with 4000 warm-up steps. Therefore, we proceeded to fine-tune the BERT embedding encoder model using only the 0.0001 learning rate. The results, presented in Table X, indicate that the optimal number of warm-up steps for fine-tuning BERT on this dataset is 4000, as it achieves the highest F1 scores across all metrics (R1, R2, and RL). Increasing or decreasing the warm-up steps results in lower model performance.

TABLE X
F1 SCORE OF BERT AFTER FINE-TUNING USING 0.0001 LEARNING RATE AND VARIOUS WARM-UP STEPS (75K TRAIN DATA)

Learning Rate	Warm-up Steps	R1	R2	RL
0.0001	1000	36.42	17.35	33.59
	25000	36.88	17.91	34.05
	4000	**37.18**	**18.19**	**34.28**
	5000	36.79	17.81	33.90
	7500	36.85	17.82	33.95
	10000	36.47	17.39	33.57

C. Comparison with Other Models

We compare our two models, BERT+CE+BPE and BERT encoder based models, with models from previous research conducted by [26] and [40]. Table XI shows the F1 score of the six models. Among the previous models, BertAbs (IndoBERT) achieved the best performance across all metrics, with the highest scores for R1 (40.94), R2 (23.01), and RL (37.89). IndoBART and BertAbs (mBERT) followed closely, while IndoGPT showed the lowest scores, particularly in R1 (37.41) and RL (31.54). Despite using much less training data (less than 40% of the original data), our proposed models achieve relatively competitive results. The BERT Encoder (75K) achieved scores of R1 (37.18), R2 (18.19), and RL (34.28), which surpassed the IndoGPT model in R1 and RL metrics despite using less than half the training data. Similarly, the BERT+CE+BPE Encoder (50K) achieved R1 (35.58), R2 (15.40), and RL (32.80), which, while slightly lower overall, demonstrates robust performance considering the smaller dataset size.

The current BERT encoder (75K train data) outperforms the BERT+CE+BPE encoder (50K train data), suggesting that increasing training data improves performance. The results indicate that with more training data, the current models might potentially match or exceed the performance of previous models.

TABLE XI
F-1 SCORE COMPARISON BETWEEN PREVIOUS AND CURRENT RESEARCH

Model	Train Data	R1	R2	RL
BertAbs (mBERT) [26]	193.9K	39.48	21.59	36.72
BertAbs (IndoBERT) [26]	193.9K	**40.94**	**23.01**	**37.89**
IndoBART [40]	193.9K	39.87	22.24	33.50
IndoGPT [40]	193.9K	37.41	20.61	31.54
Our models				
BERT+CE+BPE Encoder	50K	35.58	15.40	32.80
BERT Encoder	75K	37.18	18.19	34.28

D. Analysis of Summarization Result

The samples of the generated summaries with low and high scores are presented in Tables XII and XIII, respectively. Table XII shows that the generated summary with a low score fails to provide meaningful or relevant information. It repeats the phrase "efektif" four times, resulting in a disjointed and redundant output that lacks any substantive content or relevance to the original text. This indicates poor semantic understanding and summarization capabilities in this instance. Furthermore, the generated summary seems to take the first sentence of the document and does not contain any words from the reference summary.

In contrast, the generated summary with a high score in Table XIII is largely similar in structure and information, but it introduces a factual inconsistency by mentioning "ringotika," which appears to be an erroneous term instead of "analgetic" or another valid category. While the rest of the summary aligns closely with the human summarization, this minor error can reduce the accuracy and reliability of the model-generated summary.

IV. CONCLUSION

This research develops a model to summarize Indonesian documents using stacked embeddings as an encoder and a Transformer-based decoder. The research shows a different result from the study done by [17] regarding BERT layer selection for the model. In the summarization task, using all

TABLE XII
AN EXAMPLE OF BAD SUMMARIZATION USING BERT EMBEDDING MODEL TRAINED ON 75K DATA

Original document	*Kendati penerapan otonomi daerah sudah berlaku efektif, pemerintah pusat masih tetap membantu daerah-daerah yang mengalami bencana alam. Buktinya, dalam Tahun Anggaran 2001 pemerintah menyediakan dana bantuan sebesar Rp 400 miliar untuk menangani korban bencana di Tanah Air. Hal ini diungkapkan Menteri Permukiman dan Prasarana Wilayah Erna Witoelar di Padang, Sumatra Barat, baru-baru ini. Erna menjelaskan, kerusakan akibat bencana alam, seperti di Kabupaten Nias, Sumatra Utara dan Padang, Sumatra Barat tak saja menyusahkan korban, tapi juga menyulitkan pemerintah daerah setempat. Sebab provinsi dan kabupaten tak banyak mendapat bantuan dari pemerintah sejak Otonomi Daerah diberlakukan. Meski demikian, ungkap Erna, pemerintah tak tinggal diam. Dalam waktu dekat, lanjut Erna, bantuan untuk daerah bencana segera dikucurkan, terutama terhadap korban banjir di Padang.* [2 kalimat dengan 25 kata setelahnya tidak ditampilkan] **Translation**: Although the implementation of regional autonomy has been effectively enforced, the central government still assists regions affected by natural disasters. As proof, in the 2001 fiscal year, the government allocated aid funds amounting to Rp 400 billion to assist disaster victims across the country. This was stated by the Minister of Settlement and Regional Infrastructure, Erna Witoelar, in Padang, West Sumatra, recently. Erna explained that the damage caused by natural disasters, such as in Nias Regency, North Sumatra, and Padang, West Sumatra, not only troubled the victims but also burdened the local governments. This is because provinces and regencies have received little assistance from the government since the enactment of regional autonomy. However, Erna stated, the government has not remained idle. In the near future, continued Erna, the assistance for disaster-impacted areas would be immediately disbursed, especially for flood victims in Padang. [2 sentences with 25 words are not shown from here]
Reference summary	*Pemerintah pusat tetap membantu sejumlah daerah yang mengalami bencana alam. Departemen Permukiman dan Prasarana Wilayah menyiapkan dana sebesar Rp 400 miliar.* **Translation**: The central government continued to assist a number of impacted areas due to natural disaster. The Department of Settlement and Regional Infrastructure prepared a fund of Rp 400 billion.
Generated summary	*kendati penerapan otonomi daerah sudah berlaku **efektif efektif efektif efektif**.* **Translation**: Even though the implementation of regional autonomy had been available **effectively effectively effectively effectively**.

TABLE XIII
AN EXAMPLE OF GOOD SUMMARIZATION USING BERT EMBEDDING MODEL TRAINED ON 75K DATA

Original document	*Petugas Badan Pengawasan Obat dan Makanan (BPOM) Mataram, Nusatenggara Barat, menyita ribuan butir obat keras daftar G dari berbagai merek yang diedarkan secara ilegal. Jenis obat tersebut, antara lain antibiotik, analgetic, dan sejumlah merek lain. Hal itu diungkapkan Kepala BPOM Mataram Sriutami Ekaningtyas, baru-baru ini. Menurut Sriutami, saat disita, obat-obatan ini masih disimpan di dalam sebuah mobil boks yang biasa digunakan tersangka K. S. untuk pemasaran. Obat-obatan tersebut, menurut K. S, biasa disalurkan lewat sejumlah toko obat dan pasar obat, terutama di tempat yang terisolir. Sriutami mengakui, sebenarnya, peredaran obatan tersebut telah tercium sejak lama. Tapi, karena sulit menangkap pengedar, pemasaran obat tersebut terus berlanjut.* [2 kalimat dengan 28 kata setelahnya tidak ditampilkan] **Translation**: Officers from the Food and Drug Monitoring Agency (BPOM) Mataram, West Nusa Tenggara, confiscated thousands of pills of category G hard drugs from various brands that were being distributed illegally. These drugs included antibiotics, analgesics, and several other brands. This was revealed by the Head of BPOM Mataram, Sriutami Ekaningtyas, recently. According to Sriutami, when seized, the drugs were still stored in a box truck typically used by the suspect, K.S., for marketing. The drugs, according to K.S., were usually distributed through several drugstores and medicine markets, especially in isolated areas. Sriutami admitted that the distribution of these drugs had actually been detected for a long time. However, because it was difficult to catch the dealers, the drug marketing still continued. [2 sentences with 28 words are not shown from here]
Reference summary	*Ribuan butir obat keras daftar G dari berbagai merek yang diedarkan secara ilegal di Mataram, NTT, disita petugas BPOM. Jenis obat tersebut, antara lain antibiotik, analgetic, dan sejumlah merek lain.* **Translation**: Thousands of category G hard drugs from various brands that were being distributed illegally in Mataram, NTT, were confiscated by BPOM officers. These drugs included antibiotics, analgesics, and several other brands.
Generated summary	*petugas bpom mataram menyita ribuan butir obat keras daftar g dari berbagai merek yang diedarkan secara ilegal. jenis obat tersebut, antara lain antibiotik, **ringotika**, dan sejumlah merek lain.* **Translation**: BPOM Mataram officers confiscated thousands of category G hard drugs from various brands that were distributed illegally. These drugs included antibiotics, **ringotics**, and several other brands.

layers in BERT outperforms other layer combinations in BERT, with training and validation accuracy of 49.06 and 36.36, respectively.

The research proves that the amount of training data affects the model performance. For the models trained with 50K data, the combination of BERT, CE, and BPE results in the best performance with F1 scores of 34.17 (R1), 13.98 (R2), and 31.51 (RL). Meanwhile, for the models trained with 75K data, the stacked embeddings do not give better results than the BERT embedding. The BERT model results in the best performance F1 scores of 37.18 (R1), 18.19 (R2), and 34.28 (RL). Our proposed model achieves performance close to state-of-the-art models described in [26] and [40] despite a significant reduction in training data.

In future research, we can explore several avenues to extend this study. These include experimenting with different combinations of pre-trained embeddings, with or without BERT as the base model, and fine-tuning various parameters within the Transformer decoder. Additionally, we could investigate the impact of increasing the volume of training data or focus on improving the quality of the existing training data while maintaining a small dataset size. Each approach can enhance model performance and deepen our understanding of the model's behavior under different conditions.

REFERENCES

[1] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Systems with Applications*, vol. 165, p. 113679, 2021, doi: 10.1016/j.eswa.2020.113679.

[2] D. R. Radev, E. Hovy, and K. McKeown, "Introduction to the special issue on summarization," *Computational Linguistics*, vol. 28, no. 4, pp. 399–408, Dec. 2002, doi: 10.1162/089120102762671927.

[3] M. Jishma Mohan, C. Sunitha, A. Ganesh, and A. Jaya, "A study on ontology based abstractive summarization," in *Procedia Computer Science*, Jan. 2016, vol. 87, pp. 32–37, doi: 10.1016/j.procs.2016.05.122.

[4] M. Allahyari *et al.*, "Text summarization techniques: a brief survey," *International Journal of Advanced Computer Science and Applications*, Jul. 2017. Available: http://arxiv.org/abs/1707.02268.

[5] T. Cai, M. Shen, H. Peng, L. Jiang, and Q. Dai, "Improving transformer with sequential context representations for abstractive text summarization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Oct. 2019, vol. 11838 LNAI, pp. 512–524, doi: 10.1007/978-3-030-32233-5_40.

[6] S. Song, H. Huang, and T. Ruan, "Abstractive text summarization using LSTM-CNN based deep learning," *Multimed. Tools Appl.*, vol. 78, no. 1, pp. 857–875, 2019, doi: 10.1007/s11042-018-5749-3.

[7] I. R. Musyaffanto, G. Budi Herwanto, and M. Riasetiawan, "Automatic extractive text summarization for indonesian news articles using maximal marginal relevance and non-negative matrix factorization," in *Proceedings 2019 5th International Conference on Science and Technology, ICST 2019*, Jul. 2019, doi: 10.1109/ICST47872.2019.9166376.

[8] N. H. Gabriela, R. Siautama, C. I. A. Amadea, and D. Suhartono, "Extractive hotel review summarization based on TF/IDF and adjective-noun pairing by considering annual sentiment trends," in *Procedia Computer Science*, Jan. 2021, vol. 179, pp. 558–565, doi: 10.1016/j.procs.2021.01.040.

[9] M. R. Ramadhan, S. N. Endah, and A. B. J. Mantau, "Implementation of Textrank algorithm in product review summarization," in *ICICoS 2020 - Proceeding: 4th International Conference on Informatics and Computational Sciences*, Nov. 2020.

[10] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 280–290, doi: 10.18653/v1/K16-1028.

[11] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, Sep. 2015. Available: https://arxiv.org/abs/1409.0473v7.

[12] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proceedings of 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016*, 2016, pp. 93–98, doi: 10.18653/v1/n16-1012.

[13] J. Tan, X. Wan, and J. Xiao, "Abstractive document summarization with a graph-based attentional neural model," in *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics*, 2017, vol. 1, pp. 1171–1181, doi: 10.18653/v1/P17-1108.

[14] P. M. Hanunggul and S. Suyanto, "The impact of local attention in LSTM for abstractive text summarization," in *2nd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2019*, Dec. 2019, pp. 54–57, doi: 10.1109/ISRITI48646.2019.9034616.

[15] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, Jun. 2017, vol. 2017-Dec, pp. 5999–6009. Available: https://arxiv.org/abs/1706.03762v5.

[16] E. Egonmwan and Y. Chali, "Transformer-based model for single documents neural summarization," in *Proceedings of the 3rd Workshop on Neural Generation and Translation (WNGT 2019)*, Nov. 2019, pp. 70–79, doi: 10.18653/v1/d19-5607.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of Deep Bidirectional Transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, Oct. 2018

[18] H. Zhang, J. Xu, and J. Wang, "Pretraining-based natural language generation for text summarization," in Proceedings of *CoNLL 2019 - 23rd Conference on Computational Natural Language Learning*, *Association for Computational Linguistic*, pp. 789–797, Feb. 2019.

[19] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, pp. 3730–3740, Aug. 2019, doi: 10.18653/v1/d19-1387.

[20] S. Ren and Z. Zhang, "Pointer-generator abstractive text summarization model with part of speech features," in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, Oct. 2019, vol. 2019-Octob, pp. 514–517, doi: 10.1109/ICSESS47205.2019.9040715.

[21] Y. Iwasaki, A. Yamashita, Y. Konno, and K. Matsubayashi, "Japanese abstractive text summarization using BERT," in *Proceedings 2019 International Conference on Technologies and Applications of Artificial Intelligence, TAAI 2019*, Nov. 2019, doi: 10.1109/TAAI48200.2019.8959920.

[22] N. Khotimah and A. S. Girsang, "Indonesian news articles summarization using Genetic Algorithm," Engineering Letters, vol. 30, no. 1, pp. 152–160, 2022.

[23] D. Wardani and Y. Susanti, "Improving graph-based summarization with HTML tag and metadata features," Engineering Letters, vol. 28, no. 2, 2020.

[24] R. Adelia, S. Suyanto, and U. N. Wisesty, "Indonesian abstractive text summarization using bidirectional gated recurrent unit," in *Procedia Computer Science*, Jan. 2019, vol. 157, pp. 581–588, doi: 10.1016/j.procs.2019.09.017.

[25] R. S. Devianti and M. L. Khodra, "Abstractive summarization using genetic semantic graph for Indonesian news articles," in *Proceedings - 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, ICAICTA 2019*, Sep. 2019, doi: 10.1109/ICAICTA.2019.8904361.

[26] F. Koto, J. H. Lau, and T. Baldwin, "Liputan6: a large-scale Indonesian dataset for text summarization," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the ACL and the 10th International Joint Conference on NLP*, Nov. 2020, Accessed: Apr. 19, 2021. [Online]. Available: http://arxiv.org/abs/2011.00679.

[27] R. Wijayanti, M. L. Khodra, and D. H. Widyantoro, "Indonesian abstractive summarization using pre-trained model," in *Proceedings 3rd 2021 East Indonesia Conference on Computer and Information Technology, EIConCIT 2021*, pp. 79–84, Apr. 2021, doi: 10.1109/EICONCIT50028.2021.9431880.

[28] K. Kurniawan and S. Louvan, "IndoSum: A new benchmark dataset for Indonesian text summarization," in 2018 International Conference on Asian Language Processing (IALP), IEEE, Nov. 2018, pp. 215–220. doi: 10.1109/IALP.2018.8629109.

[29] B. Heinzerling and M. Strube, "Sequence tagging with contextual and non-contextual subword representations: a multilingual evaluation," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, A. Korhonen, D. Traum, and L. Màrquez,

Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 273–291. doi: 10.18653/v1/P19-1027.
[30] A. S. Fadel, M. E. da. A. Saleh, and O. A, "Arabic aspect extraction based on stacked contextualized embedding with deep learning," IEEE Access, vol. 10, pp. 30526–30535, 2020, doi: 10.1109/ACCESS.2022.3159252.
[31] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), K. Erk and N. A. Smith, Eds., Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725.
[32] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, K. Knight, A. Nenkova, and O. Rambow, Eds., San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 260–270. doi: 10.18653/v1/N16-1030.
[33] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.
[34] L. Tanoto, "Indonesian abstractive text summarizer model using the combination of pre-trained embeddings and transformer decoder," Undergraduate thesis, Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia, 2022.
[35] B. Wilie et al., "IndoNLU: benchmark and resources for evaluating Indonesian natural language understanding," in Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, K.-F. Wong, K. Knight, and H. Wu, Eds., Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 843–857.
[36] B. Heinzerling and M. Strube, "BPEmb: tokenization-free pre-trained subword embeddings in 275 languages," in Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), N. C. (Conference chair), K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, Eds., Miyazaki, Japan: European Language Resources Association (ELRA), May 2018.
[37] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.
[38] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "FLAIR: An easy-to-use framework for state-of-the-art NLP," in 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), 2019, pp. 54–59.
[39] K. Ganesan, "ROUGE 2.0: updated and improved measures for evaluation of summarization tasks," *arxiv.org/abs/1803.01937*, Mar. 2018. Available: https://arxiv.org/abs/1803.01937.
[40] S. Cahyawijaya et al., "IndoNLG: benchmark and resources for evaluating indonesian natural language generation," in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 8875–8898. doi: 10.18653/v1/2021.emnlp-main.699.