HASumRuNNer: An Extractive Text Summarization Optimization Model Based on a Gradient-Based Algorithm

Muljono [1,*], Mangatur Rudolf Nababan [2], Raden Arief Nugroho [3], and Kevin Djajadinata [1]

[1] Department of Informatics Engineering, Universitas Dian Nuswantoro, Semarang, Indonesia;
Email: p31201902233@dinus.ac.id (K.D.)
[2] English Department, Universitas Sebelas Maret, Surakarta, Indonesia;
Email: amantaradja.nababan_2017@staff.uns.ac.id (M.R.N.)
[3] English Department, Universitas Dian Nuswantoro, Semarang, Indonesia;
Email: arief.nugroho@dsn.dinus.ac.id (R.A.N.)
*Correspondence: muljono@dsn.dinus.ac.id (M.)

***Abstract*—This article is based on text summarization research model, also referred to as "text summarization", which is the act of summarizing materials in a way that directly communicates the intent or message of a document. Hierarchical Attention SumRuNNer (HASumRuNNer), an extractive text summary model based on the Indonesian language is the text summary model suggested in this study. This is a novelty for the extractive text summary model based on the Indonesian language, as there is currently very few related research, both in terms of the approach and dataset. Three primary methods—BiGRU, CharCNN, and hierarchical attention mechanisms—were used to create the model for this study. The optimization in this suggested model is likewise carried out using a variety of gradient-based methods, and the ROUGE-N approach is used to assess the outcomes of text synthesis. The test results demonstrate that Adam's gradient-based approach is the most effective for extracting text summarization using the HASumRuNNer model. As can be seen, the values of RED-1 (70.7), RED-2 (64.33), and RED-L (68.14) are greater than those of other methods employed as references. The approach used in the suggested HASumRuNNer Model, which combines BiGRU with CharCNN, can result in more accurate word and sentence representations at word and sentence levels. Additionally, the word and sentence-level hierarchical attention mechanisms aid in preventing the loss of information on each word in documents that are typically brought on by the length of the input model word or sentence.**

***Keywords*—extractive text summarization, hierarchical attention mechanism, deep learning, BiGRU, CharCNN**

I. Introduction

Compacting a written document to make the core concept obvious is the process of text summarization [1]. Text summarization falls into two categories: abstractive and extractive. Since an abstractive text summarizing uses word-for-word paraphrasing, some words in the summary may not be present in the original text. By paraphrasing the text sentence by sentence, extractive text summarization ensures that the words used are those of the original text. There are various approaches to text summarization. Some of the fundamental methods used are unsupervised, supervised, and neurally supervised. Studies have shown that text summarization has recently shifted more in the direction of neural-supervised learning [2–4].

Text summarization research for Indonesian documents is still relatively rare, with few publicly available datasets. Finally, as a result of research conducted by Andrearczyk *et al.* [5], a public dataset of Indonesian for text summarization known as Indosum was published. Furthermore, text summarization methods in Indonesia are still limited to general methods based on Recurrent Neural Network (RNN), Gated Recurrent Units (GRU), and Long Short-Term Memory (LSTM), with little exploration as in Ref. [2] and [6]. Based on these facts, the author conducts research on text summarization in Indonesian using the Hierarchical Attention SumRuNNer method or model (HASumRuNNer). The studies of [4, 7, 8] serve as the foundation for proposing this model. In the first study, Bhargava *et al.* [7] proposed a model for extractive text summarization called NeuralSum. For encoder-level sentences, the model employed CharCNN for word-level encoding and LSTM with an attention mechanism. CharCNN was used in this model because it was effective in classifying text at the sentence level, such as for sentiment analysis [9]. Then, to reduce the risk of vanishing gradient problems [10], LSTM was chosen, and an attention mechanism was added to increase focus and reduce information loss in long word and sentence sequences, the authors of proposed a model called SumaRuNNer in the second study [8]. The model used BiGRU [11] with two levels, the first layer served as the word level and the second layer served as the sentence

Manuscript received November 30, 2022; revised February 2, 2023, accepted March 23, 2023; published July 11, 2023.

level. The features for content, salience, novelty, absolute positional embedding and relative positional embedding are then introduced during the final classification stage. By including these features, the model was able to comprehend the context and sentence structure and anticipate sentences for text summarization better than other models.

Both studies still employed single method, either CharCNN or BiGRU, to turn word representations in documents into sentences, notably at the word level. In the first study, the attention mechanism was only used at the phrase level, and it was not used at all in the second study. While in a document, a correct word representation is also crucial to identify which sentences are part of text summarization. There is a hazardous possibility that the information at the word level cannot be retained when it is acquired at the sentence level. As a result, the proposed HASumRuNNer model by the authors can change and supplement the two models used in the study.

Hierarchical attention in this model uses references from the third research model, CRHASum [4]. The use of hierarchical attention in this model helps obtain important information on every word or sentence in the document. The HASumRuNNer model uses two methods for the processing at the word level. The first method is BiGRU, which adds an attention mechanism as in the CRHASum model. The second method employs CharCNN, as in the NeuralSum model. The outputs from the two methods are combined to become the input at the sentence level. The sentence level again uses the BiGRU method and the same attention mechanism as in the CRHASum model. The sentence-level output is used for the final classification using the Summar-RuNNer model's technique, which includes salience addition, absolute position, relative position information, and replacement of novelty information with a combination of the previous and following context generated by the attention mechanism in the sentence-level.

A better representation of the words and sentences in the document can be obtained by using two methods at the word level. This is because there are two processes used to create the weights for the word level, which should, in principle, result in the addition of information to each sentence and enhance the process of summarizing the text accurately. The HASumRuNNer model is examined utilizing several gradient-based techniques during the backpropagation process, in addition to concentrating on merging two methods and hierarchical attention. According to researches [12–14], the application of various gradient-based approaches yields various outcomes and degrees of accuracy in neural-based models. Additionally, the Indosum dataset is utilized to test this model. This Indosum dataset is selected since it has been utilized in Ref. [4], making it a direct benchmark or baseline for the author's future research.

The contribution of this research is later proposed to the HASumRuNNer model, which can produce more accurate word representations with the feature processing of two methods, namely CharCNN and BiGRU, and the use of hierarchical attention mechanisms to prevent information loss in documents with long sentences or words, so that it can help people in texting extractive summarization with greater accuracy. The following is the research's next section: A summary of some related research is provided in Section II. The gradient-based technique employed in this study and the proposed model's methodology are both explained in Section III of the paper. In Section IV, we provide a more thorough explanation of the dataset, the parameter that the suggested model relied upon, and the outcomes and comparisons of the text summarization for each gradient-based approach. Conclusion and possible future research, Section V is presented in the final part.

II. Related Works

Text summarization has been the subject of numerous studies, particularly when utilizing the deep learning approach. From Andrearczyk *et al.* [5] comes the first one. The absence of datasets for text summarizing in Indonesian is the basis for the study. Indosum is the name of a dataset published in Indonesian. Twenty thousand documents make up this dataset, which is divided into six sections: entertainment, motivation, sport, showbiz, headlines, and technology. This study published the dataset and tested many approaches, including oracle-based, unsupervised, non-neural supervised, and neural supervised. Then, we employed ROUGE with types R-1, R-2, and R-L for the evaluation procedure. The ROUGE value is higher when using a neural supervised learning approach, especially Neuralsum with a word embedding of 300. However, the authors stated that the resulting ROUGE value is still far from the maximum possible. This indicates that there is still a lot of room for improvement with this Indosum dataset.

The following one, from Bhargava *et al.* [7], proposes a supervised neural-based model called Neuralsum. The model used the CharCNN approach to represent sentences at the sentence level. The LSTM technique was then used to process each sentence representation, together with the attention mechanism. Based on the outcomes, sentences used in the text summaries were selected. Model testing was carried out using the DUC 2002 and DailyMail datasets. The evaluation procedure then used types R1, R-2, and R-L together with the ROUGE method. According to the findings, the URANK technique produced the highest R-1 value in the DUC 2002 dataset, the TGRAPH method produced the highest R-2 value, and the Neuralsum method with the Sentence Extractor type produced the top R-L value. Then, using the Neuralsum method and the Sentence Extractor type, all of the top R1, R-2, and R-L values for the DailyMail dataset were obtained.

A model called SummaRuNNer was proposed in a different study by R. Bhargava [8]. Both BiGRU at the word level and BiGRU at the sentence level made up the model. To extract additional features such as content, salience, novelty, absolute position, relative position, and bias, the output from the sentence level was further processed. To determine which sentence was a part of the text summarization, all of these attributes were combined. The DUC 2002 and DailyMail datasets were used in this

study, and the ROUGE method types R-1, R-2, and R-L were used in the evaluation procedure. The extractive and abstractive text summarization were done using the SummaRuNNer model, which was being developed. The extractive SummaRuNNer approach produced the highest R-1 and R-2 values in the DailyMail dataset, while the Lead-3 technique produced the highest R-L value. The highest R-1 value was then obtained by the URANK approach, the highest R-2 value was the TGRAPH method, and the highest R-L value was obtained by the Cheng and Lapata's method for the DUC 2002 dataset with a text summary of 75 words [10]. This is because the existing SummaRuNNer model was trained using a DailyMail dataset, which had a different domain than the DUC 2002 dataset. According to the author of this study, the proposed SummaRuNNer model is more interpretable due to the addition of information and features, such as content, salience, and novelty.

The research by Anand *et al.* [4] employed the hierarchical attention mechanism and proposed the CRHASum model. This model was capable of picking up on semantic contextual information and the connections between document aspects. For each existing sentence, the existing model used a hierarchical attention mechanism to add information at the word and sentence levels. The BiGRU approach with hierarchical attention at the word level and sentence level made up the CRHASum model itself. The sentence-level output underwent additional processing to obtain the prior and subsequent context vector attributes. The process of selecting sentences that was part of the text summarization itself used the original output from the sentence-level, previous and following context, as well as additional features such as time features, topic features, sentiment features, and statistical features. Then, the dataset used to test the model was DUC 2001, DUC 2002, and DUC 2004 with the ROUGE evaluation method type R-1 and R-2. The results showed that the CRHASum method with Semantics Feature (SF) obtained the highest R-1 and R-2 values, only losing to the Upper bound method for every dataset used in this research. This shows that the use of semantic features helped the model learn the representation and context of each word in the document. Besides that, hierarchical attention was also an effective method to be used in the model by helping the model add or enrich the context of each word in the document.

The African Vulture Optimization Algorithm (AVOA) by Shaddeli [15] also related to others studies that are cited. The simulation results demonstrated that the proposed BAOVAH algorithm outperformed other binary meta-heuristic algorithms in terms of performance. It also performed well in terms of feature selection. Moreover, Hosseinalipour *et al.* [16] used two distinct wrapped feature selection procedures based on the Farmland Fertility Algorithm (FFA) for the selection of research features. The FFA algorithm was suggested in two binary forms, BFFAS and BFFAG. The findings indicated that the suggested strategy outperformed competing approaches in terms of classification accuracy, the average number of features selected, and objective function value. Similar feature selection research was done by Shaddeli [17]. A binary hyper-heuristic feature ranking method was created in this research to address the feature selection issue. The outcomes demonstrated that the BFRA algorithm behaved in low dimensions like a robust meta-heuristic algorithm. Maragheh *et al.* [18] carried out pertinent research. This research proposed a new model for Managed Long-Term Care (MLTC) based on the LSTM network and Spotted Hyena Optimizer-Long Short-Term Memory (SHO-LSTM). According to the evaluation, the suggested model was more accurate than LSTM, Genetic Algorithm LSTM (GA-LSTM), Particle Swarm Optimization LSTM (PSO-LSTM), Artificial Bee Colony LSTM (ABC-LSTM), Harmony Algorithm Search LSTM (HAS-LSTM), and Differential Evolution LSTM (DE-LSTM). The other related works in text summarization are explained in Table I below.

TABLE I. RELATED WORKS IN TEXT SUMMARIZATION.

Author	Methods	Dataset	Evaluation Method
[7]	NeuralSum (CharCNN + LSTM)	DUC 2002, DailyMail	ROUGE (R-1, R-2, R-L)
[8]	SummaRuNNer (BiGRU + content, salience, novelty feature)	DUC 2002, CNN News, DailyMail	ROUGE (R-1, R-2, R-L)
[19]	Attentive Encoder + RNN (Unidirectional & Bidirectional)	CNN News	ROUGE (R-1, R-2, R-4, R-L)
[5]	Oracle, Lead3, Sumbasic, LSA, Lexrank, Textrank, Bayes, HMM, Maxent, Neuralsum (CharCNN + LSTM)	Indosum	ROUGE (R-1, R-2, R-L)
[20]	RBM Network + Fuzzy Logic	Kaggle News	ROUGE (Unknown)
[21]	LSA + Self Organizing Map (SOM) + ANN	Opinosis	ROUGE (R-1, R-2, R-L, R-SU4)
[22]	TreeLSTM	CLWritten, CLSpoken	Simple String Accuracy (SSA), Compression Rate, F-1 Score
[6]	BiGRU	Wikipedia	ROUGE (R-1, R-2, R-L)
[4]	CRHASum (Hierarchical Attention + BiGRU + Semantic Feature)	DUC 2001, DUC 2002, DUC 2004	ROUGE (R-1, R-2)
[2]	BiGRU	Indonesian Journal Documents	ROUGE (R-1, R-2)
[3]	CharCNN + BiLSTM	CNN News, DailyMail	ROUGE (R-1, R-2, R-L)
[23]	CNN + LSTM	Indian Supreme Court Judgements	ROUGE (R-1, R-2, R-L)
[24]	Generative Adversarial Networks (GRU + RNN)	Multiling 2015	ROUGE (Unknown)

[25]	Auto Encoder (AE) + VAE + ELM-AE	SKE	ROUGE (R-1, R-2)
[26]	CNN	Multiling 2015	F-1 Score
[27]	BERTSUM	CNN News, DailyMail, NYT, XSum	ROUGE (R-1, R-2, R-L)
[28]	BERT + GPT-2	COVID-19 Open Research Dataset Challenge	ROUGE (R-1)
[29]	Two Stage Encoder Model (TSEM)	CNN News, DailyMail	ROUGE (R-1, R-2, R-L)
[30]	DT+CM+NB	ISEAR, Sentiment polarity v2.0, SST, Yelp, IMDb	Accuracy, Sensitivity (recall), Specificity, Precision, F-measure, MCC

In this research, we proposed a model namely HASumRuNNer for extractive text summarization using Indonesian dataset. The model consists of three main methods, they are BiGRU, CharCNN, and hierarchical attention mechanism. Besides that, for more precise determination of sentences, we add the summarization, value from salience, absolute position, relative position information, previous context and following context in the final classification.

III. Methodology

The stages of this research are described in the flowchart in Fig. 1 and the suggested HASumRuNNer model is thoroughly detailed at the word and sentence levels in this section. In addition, the gradient-based algorithm and HASumRuNNer model evaluation procedure are also described.

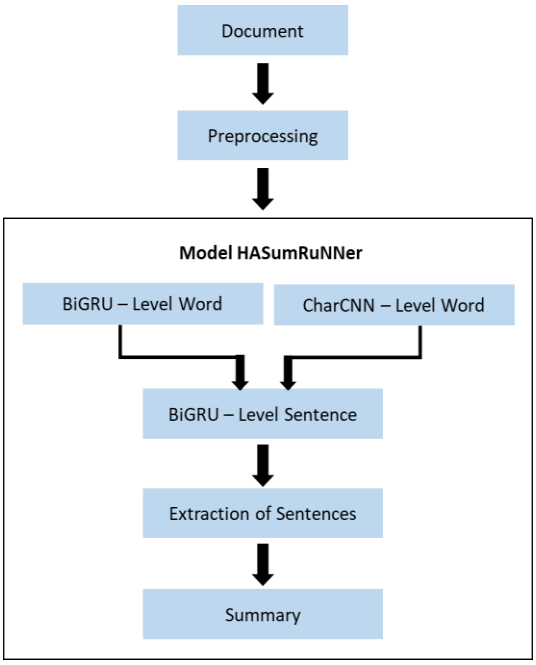

Figure 1. Flowchart of the proposed method.

The following are the steps of this study:

1. Creating research's dataset: documents are gathered, initially in the form of text.
2. Preprocessing: the process of converting all of the words in a document into basic words and removing punctuation marks and stop words. Following that, word representations in the form of integers are created for each word using the FastText library.
3. At this point, the process splits into two steps, namely:
   a. BiGRU-Level Word where the word representation created in step 2 is processed at the BiGRU layer along with producing sentence representations using hierarchical attention.
   b. CharCNN-Level Word where sentence representations are created by processing the word representation created in step 2 at the CharCNN layer.

To represent the real sentence, the outcomes of procedures a and b are concatenated.

4. Employing BiGRU-Level Sentence: in the BiGRU layer, which likewise employs hierarchical attention to produce the final output of the sentence, each existing sentence representation is processed.
5. Extracting sentences: The output no. 4 findings is utilized to determine each sentence's substance, salience, attention, and absolute and relative position values. The probability that a sentence is included in the summary or not is calculated by adding together all of these numbers.
6. Summarizing: each sentence that is a part of the summation is chosen for the summary and can finally be used to summarize the original content.

A. *HASumRuNNer*

1) *Word-level*

In this word-level, two types of models are used, they are Bi-GRU and CharCNN.

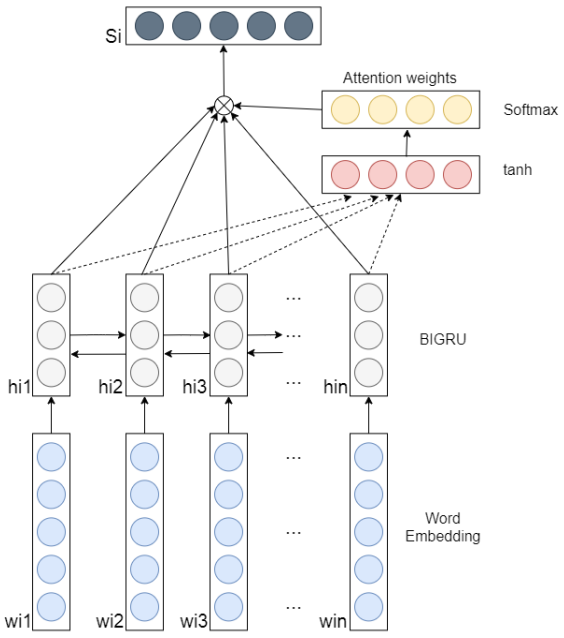

Figure 2. BiGRU layer at word-level.

Based on Fig. 2, the form of the Bi-GRU model to be built is similar to what is used by Alami and Meknassi *et al.* [3]. This model has a single bi-GRU layer and accepts word input from every sentence in the document. Each word is also represented in a numeric vector, a process known as "word embedding". The FastText library is used to generate the word embedding. Following that, the Bi-GRU layer encodes each of the word, generating a hidden state (h) of the word and combining it to form a single sentence (S_i) representation in a document. The formula of the beginning of word input process to becoming a sentence is as follows.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$\tilde{h}_t = tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t \quad (1)$$

Eq. (1) is applied to the existing Bi-GRU layer, where *t* is a timestep, or in this case, 1 timestep = 1 word, and x_t is the embedding word of the existing word. The hidden state or encoded result of the existing word is then h_t. The symbols *U* and *W* represent the hidden state weights, and each of the input word and *b* are biased.

$$\vec{h}_t = GRU_{ltr}(x_t, \vec{h}_{t-1})$$
$$\overleftarrow{h}_t = GRU_{rtl}(x_t, \overleftarrow{h}_{t-1})$$
$$h_t = [\vec{h}_t; \overleftarrow{h}_t]. \quad (2)$$

Then, because the Bi-GRU layer is used, there are two hidden states: forward hidden and backward hidden. The two hidden states are concatenated to form the final hidden state of the existing word.

$$u_t = (V_w)^\top tanh\,(W_w[h_t] + b_w)$$
$$\alpha_{ti} = softmax(u_t) = \frac{\exp(u_{ti})}{\sum_j \exp(u_{tj})}$$
$$j_t = \sum_i \alpha_{ti} h_i. \quad (3)$$

Each hidden state of the word must be combined and added hierarchically to an attention layer to become a sentence representation that can be used at the sentence level. The combining formula is shown above. The symbol a_{ti} represents the t_{-th} sentence in the i_{-th} word's attention value. The formula for calculating attention value is softmax on u_t, where *t* is the word *t*. After calculating all of the attention values for each word, they can be combined into a sentence representation (j_t) by summing the multiplication results between the attention value and the hidden state. The j_t value in this case represents a sentence representation of the Bi-GRU method.

The following is for sentence representation of the CharCNN method. The CharCNN method is more or less the same as in Fig. 3.

Every word in the sentence goes through a word embedding process as in the previous Bi-GRU model. After that, each sentence and word in the sentence carry out a convolutional process with the following Eq. (4).

$$f_j^i = \tanh\,(W_{j:j_{+c-1}} \otimes K + b) \quad (4)$$

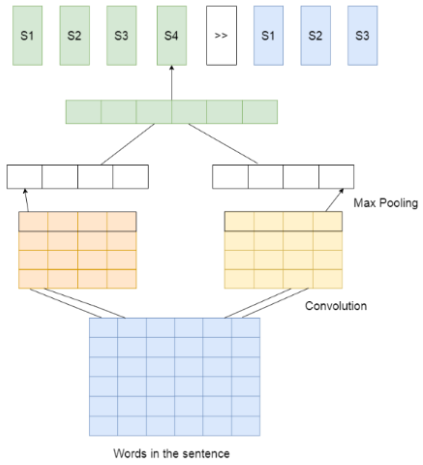

Figure 3. CharCNN layer at word-level.

The f_j^i symbol represents the feature map for the *i*th sentence in the *j*th sequence. To get this feature map, multiplication (dot product) is carried out between *W* which represents 1 sentence (has size $n \times d$ where *n* is the maximum number of words in the sentence and *d* is the size / the length of the word embedding) with *K*, where *K* is the kernel size with a width of *c*. Furthermore, in each feature map, max pooling is carried out with the formula below to finally produce a sentence representation from the CharCNN method.

$$S_i, K = \max f_j^i \quad (5)$$

The results of sentence representations from the Bi-GRU and CharCNN methods are combined (concatenation) in order to be used at the sentence-level.

2) Sentence-level

*I*n this sentence-level, a model that is almost similar to the one conducted in the research by Alami [3] is used again.

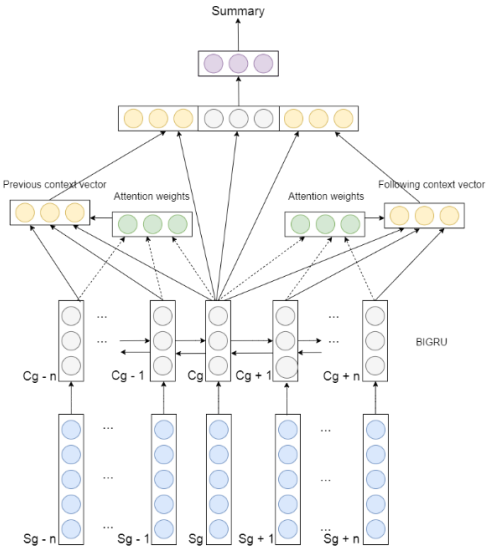

Figure 4. BiGRU layer at sentence-level.

The model is composed of one bi-GRU layer and one hierarchical attention layer showed in Fig. 4. For the Bi-GRU formula, it is still the same at the word level, while the formula for the attention layer is as follows:

$$u_t = (V_w)^{\top} \tanh\,(W_w[c_t{:}\, c_g] + b_w)$$

$$\alpha_{ti} = softmax(u_t) = \frac{\exp\,(u_{ti})}{\sum_j \exp\,(u_{tj})}$$

$$a_t = \sum_i \alpha_{ti} c_i \quad (6)$$

The c_t symbol is the sentence representation from the word-level, while c_g is the hidden state of the sentence after it has been encoded by the BiGRU layer. The symbol α_{ti} again represents the attention value for the *t*-th sentence in a document, and the i value here is for the *i*-th sentence in the range 1 to *j* (previous) or in the range n (number of sentences) to *j* (following) where *j* = *t*. For example, the total sentence in the document is 5, so the previous attention value from the 3rd sentence has a value of *t* = 3, *j* = 3, and *i* in the *i* range from 1 to 3, while the attention value following the third sentence has the value of *t* = 3, *j* = 3, and *i* in the range of *i* ranging from 5 to 3. After the attention value for each sentence is calculated, we can find the value of a_t which is the multiplication of the attention value and the hidden state of the sentence. The a_t value is obtained from two sides, namely the previous and following. The two values are finally added up.

Then, the summarization process is based on the number of summarization sentences that users want to produce. If users want to get five sentences to be part of the summarization, then choose the five sentences that have the highest value generated using Eq. (7).

$$d = tanh\left(W_d \frac{1}{N_d} \sum_{j=1}^{N^d} [\vec{h}_j; \overleftarrow{h}_j] + b\right)$$

$$\begin{aligned} P(y_j = 1 | h_j, p_j, a_j, d) = \sigma(& W_c h_j && \#\,(content) \\ & + h_j^{\top} W_s d && \#\,(salience) \\ & + W_a a_j && \#\,(\text{attention}) \\ & + W_{ap} p_j^a && \#\,(\text{abs. pos. imp.}) \\ & + W_{rp} p_j^r && \#\,(\text{rel. pos. imp.}) \\ & + b) && \#\,(\text{bias}) \end{aligned} \quad (7)$$

In Eq. (7), *d* is the representation of the document derived from multiplying the weight with the average hidden state of all sentences plus the bias. Then, to predict the value of 0/1, we use the values of content, salience, attention, the absolute and relative position of the sentence in an embedding document, and also bias.

Furthermore, the existing model is tested with various gradient-based algorithms like SGD, Adagrad, RMSProp, Adam, Adadelta, Adamax, and Nadam. The results obtained by each of the algorithms above are analyzed to see which one gives the best performance.

B. Gradient Based Algorithm

Gradient descent is a method for determining a function's local (or, better yet, global) minimum. Gradient descent is a technique commonly used in deep learning-based models to reduce errors in existing models by updating variables such as weight and bias. The update occurs during the backpropagation model or, if the model is recurrent, during the Back Propagation Through Time (BPTT) process. After all training data have been processed, the updating process is carried out using standard gradient descent. The update process is carried out in Stochastic Gradient Descent (SGD) every time one set of training data is processed, or every time one batch of training data is processed. One batch can consist of more than one training dataset, so when using a batch, the update process can occur as many times as the number of training datasets divided by the batch size. For example, if there are 100 training datasets and the batch size is 3, the update can happen 34 times. A common issue with gradient descent is it often stops in a local minimum that is far from the global minimum [29]. However, several modifications of gradient descent have been made to overcome these problems, including Nesterov [31], Adam (adaptive moment estimation) [32], Adagrad (adaptive gradient)[33], and Adadelta (adaptive learning rate) [34]. Some examples of the basic formulas of the existing gradient descent methods are as follows:

1) Stochastic gradient descent (SGD)

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t} \quad (8)$$

Information:

w_{t+1}: weight / parameter to be updated / optimized during timestep t + 1

w_t: weight / parameter that you want to update / optimize during timestep t

α: the learning rate which we usually define directly in the range > 0 to 1

$\frac{\partial L}{\partial w_t}$: gradient of the loss function which we want to minimize the value

2) AdaGrad

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \frac{\partial L}{\partial w_t}$$

$$v_t = v_{t-1} + \left[\frac{\partial L}{\partial w_t}\right]^2 \quad (9)$$

Information:

w_{t+1}: weight/parameter to be updated/optimized during timestep *t* + 1

w_t: weight/parameter that you want to update/optimize during timestep *t*

α: the learning rate which we usually define directly in the range > 0 to 1

$\frac{\partial L}{\partial w_t}$: gradient of the loss function which we want to minimize the value

v_t: the sum of all gradient values from the loss function (which has been raised to the power) until timestep *t* (the value of v is initialized with the value 0 when the timestep is 0)

v_{t-1}: the sum of all gradient values from the loss function (which has been raised to the power) until timestep *t* – 1

ϵ: epsilon to prevent division by 0, the value of epsilon that is usually used is very small, for example 10^{-7}

3) RMSProp

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \in}} \frac{\partial L}{\partial w_t}$$

$$v_t = \beta v_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial w_t}\right]^2 \quad (10)$$

Information:
w_{t+1}: weight/parameter to be updated/optimized during timestep $t + 1$
w_t: weight/parameter that you want to update/optimize during timestep t
α: the learning rate which we usually define directly in the range > 0 to 1
$\frac{\partial L}{\partial w_t}$: gradient of the loss function which we want to minimize the value
v_t: the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep t (the value of v is initialized with the value 0 when the timestep is 0)
v_{t-1}: the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep $t - 1$
$\in$: epsilon to prevent division by 0, the value of epsilon that is usually used is very small, for example, 10^{-6}
β: constant value, where the value used is 0.9 (as suggested by the author of this method)

4) Adam

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t + \in}} \hat{m}_t$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w_t}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2 \quad (11)$$

Information:
w_{t+1}: weight/parameter to be updated/optimized during timestep $t + 1$
w_t: weight/parameter that you want to update/optimize during timestep t
α: the learning rate which we usually define directly in the range > 0 to 1
$\frac{\partial L}{\partial w_t}$: gradient of the loss function which we want to minimize the value
m_t: the sum of all values of the exponential moving average gradient from the loss function (which are not raised to a power) until timestep t (the value of m is initialized with the value 0 when the timestep is 0)
m_{t-1}: the sum of all exponential moving average gradient values from the loss function (which are not raised to the power) until timestep $t - 1$
v_t: the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep t (the value of v is initialized with the value 0 when the timestep is 0)
v_{t-1}: the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep $t - 1$
$\in$: epsilon to prevent division by 0, the value of epsilon commonly used is very small, eg., 10^{-8} (as suggested by the creator of this method)
β_1: constant value for the variable m, where the value used is 0.9 (as suggested by the author of this method)
β_2: constant value for the variable, where the value used is 0.999 (as suggested by the author of this method)
$\hat{m}_t$ and $\hat{v}_t$: function as bias corrections for variables m and v

5) Adadelta

$$w_{t+1} = w_t - \frac{\sqrt{D_{t-1} + \in}}{\sqrt{v_t + \in}} \frac{\partial L}{\partial w_t}$$

$$D_t = \beta D_{t-1} + (1-\beta)[\Delta w_t]^2$$

$$v_t = \beta v_{t-1} + (1-\beta)[\frac{\partial L}{\partial w_t}]^2$$

$$\Delta w_t = w_t - w_{t-1} \quad (12)$$

Information:
w_{t+1}: weight/parameter to be updated/optimized during timestep $t + 1$
w_t: weight/parameter that you want to update/optimize during timestep t
$\frac{\partial L}{\partial w_t}$: gradient of the loss function which we want to minimize the value
v_t: the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep t (the value of v is initialized with the value 0 when the timestep is 0)
v_{t-1}: the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep $t - 1$
D_t: the sum of all exponential moving average values from the difference/delta weight (which has been raised to the power) until timestep t (the value of D is initialized with the value 0 when the timestep is 0)
D_{t-1}: the sum of all exponential moving average values from the difference/delta weight (which has been raised to the power) until timestep $t - 1$
$\in$: epsilon to prevent division by 0, the value of epsilon that is usually used is very small, for example, 10^{-6}
β: constant value, where the value used is 0.95 (as suggested by the author of this method)

6) Adamax

$$w_{t+1} = w_t - \frac{\alpha}{v_t} \widehat{m_t}$$

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$$

$$m_t = \beta_1 - m_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t}$$

$$v_t = \max\left(\beta_2 v_{t-1}, \left|\frac{\partial L}{\partial w_t}\right|\right) \quad (13)$$

Information:
w_{t+1}: weight/parameter to be updated/optimized during timestep $t + 1$
w_t: weight/parameter that you want to update/optimize during timestep t
α : the learning rate which we usually define directly in the range > 0 to 1
$\frac{\partial L}{\partial w_t}$: gradient of the loss function which we want to minimize the value
m_t: the sum of all values of the exponential moving average gradient from the loss function (which are not raised to the power) until timestep t (the value of m is initialized with the value 0 when the timestep is 0)
m_{t-1}: the sum of all exponential moving average gradient values from the loss function (which are not raised to the power) until timestep to $t - 1$
v_t: the maximum value of the exponential moving average gradient from the loss function (which has been normalized) until timestep t (the value of v is initialized with the value 0 when the timestep is 0)
v_{t-1} : maximum exponential moving average gradient from loss function (which has been normalized) until timestep $t - 1$
$\in$: epsilon to prevent division by 0, the value of epsilon that is usually used is very small, for example, 10^{-7}
β_1: constant value for the variable m, where the value used is 0.9 (as suggested by the author of this method)
β_2: constant value for the variable v, where the value used is 0.999 (as suggested by the author of this method)
$\widehat{m}_t$: serves as bias corrections for the variable m

7) Nadam

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\widehat{v_t}}+\in}(\beta_1\widehat{m_t} + \frac{1-\beta_1}{1-\beta_1^t}\frac{\partial L}{\partial w_t})$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w_t}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2 \quad (14)$$

Information:
w_{t+1}: weight/parameter to be updated/optimized during timestep $t + 1$
w_t: weight/parameter that you want to update/optimize during timestep t
α: the learning rate which we usually define directly in the range > 0 to 1
$\frac{\partial L}{\partial w_t}$: gradient of the loss function which we want to minimize the value
m_t: the sum of all values of the exponential moving average gradient from the loss function (which are not raised to a power) until timestep t (the value of m is initialized with the value 0 when the timestep is 0)
m_{t-1} : the sum of all exponential moving average gradient values from the loss function (which are not raised to the power) until timestep $t - 1$
v_t : the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep t (the value of v is initialized with the value 0 when the timestep is 0)
v_{t-1} : the sum of all exponential moving average gradient values from the loss function (which has been raised to the power) until timestep $t - 1$
$\in$: epsilon to prevent division by 0, the value of epsilon that is usually used is very small, for example, 10^{-7}

C. Evaluation

Following the training and testing with the aforementioned model, the measurements of the existing text summarization predictions can be made using the ROUGE-N method. A higher ROUGE value indicates a more accurate summarization result.

$$ROUGE - N = \frac{\sum_{S\in S_H}\sum_{g_n\in S} C_{match}(g_n)}{\sum_{S\in S_H}\sum_{g_n\in S} C(g_n)} \quad (15)$$

where S_H is the total number of manual summaries, S is 1 individual in the manual summary, g_n is the specified N-gram, and C (g_n) is the number of co-occurrences of g_n in the manual summary and automatic summary.

IV. Result and Discussion

This section consists of information from the preprocessing dataset, used model parameters, experiment result from the model, and model evaluation analysis.

A. Preprocessing Dataset

Preprocessing is done first on the existing Indosum dataset in this study. The dataset contains 18,774 records and is intended for text summarization in Indonesian. It is divided into six categories: entertainment, inspiration, sport, showbiz, headline, and technology. The preprocessing procedure consists of the following steps:

1. Tokenization: the process of converting each sentence in a document into a group of words. The sentence "Today Ani is playing together with his friends." is broken down into a group of words that include "day", "this", "Ani", "play", "together", "with", and "friends".
2. Stop Word Removal: the process of removing common words that appear frequently but have no meaning, such as "this", "with", "and", and other conjunctions.
3. Stemming: the process of determining the root word from each tokenized word for example, the word "playing" is changed to the root word, which is "play", and the word "friendly" becomes "friend". In this research, the stemming process for each word in the document is done by using a Python library called Sastrawi.

4. Word Embedding: this process is used to generate a numerical matrix representation for each word that has passed through the previous three processes. This study employs a matrix of 100 numbers to represent one word. The FastText Python library is used to generate the matrix.

The numbers of sentences and words used in this study are limited to one document with a maximum length of 15 sentences and a maximum length of 25 words for each sentence. Based on these constraints, one document will be represented by a matrix with the dimensions of 15 × 25 × 10.

B. *Model Parameters*

The HASumRuNNer model used in this study has a shape shown in Fig. 5. Initially, the model accepts input that has the size of the numbers of documents × 5 sentences × 25 words × 100 numbers that represent one word. Then, it enters the word-level where the flow is divided into 2, namely CharCNN and BiGRU.

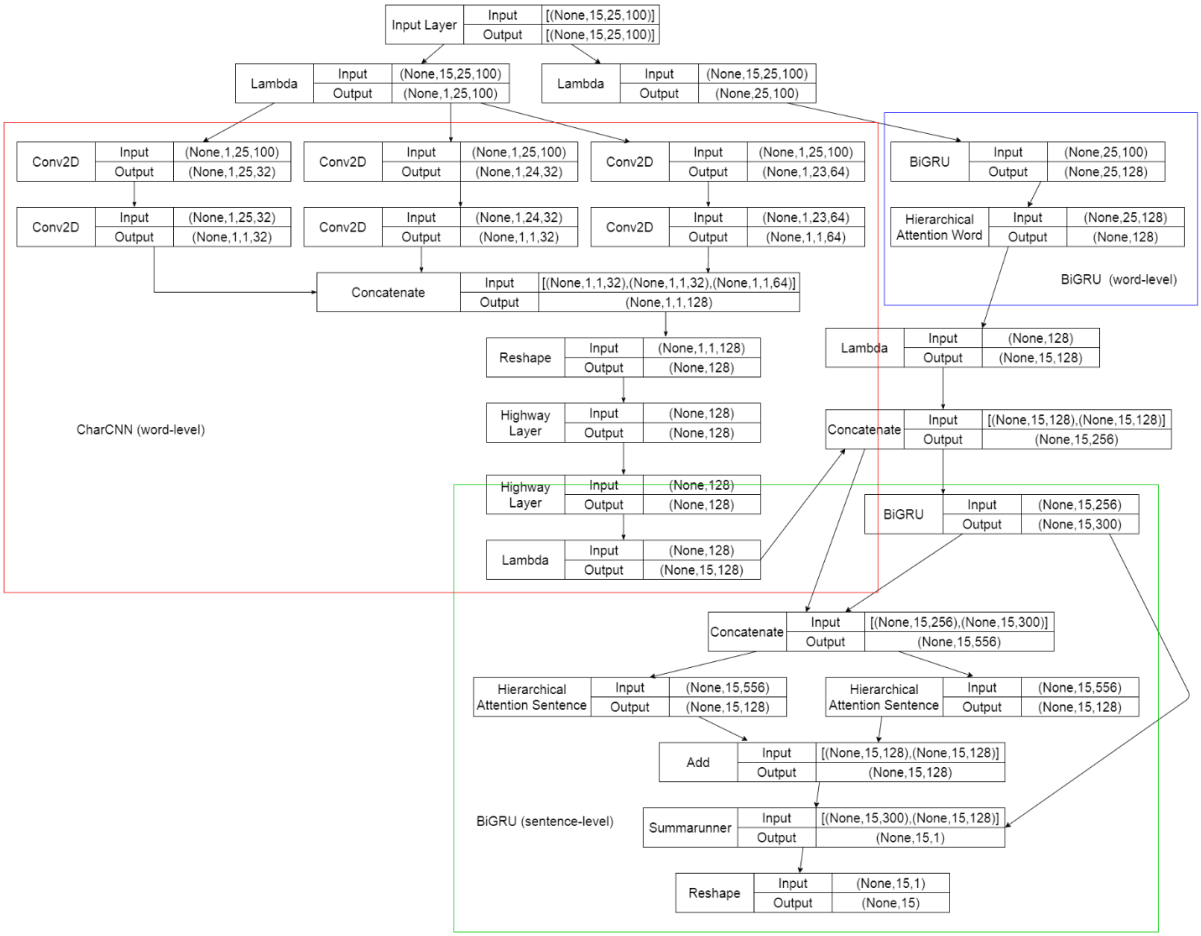

Figure 5. HASumRuNNer model parameters.

1. The input is transformed on the CharCNN side, where the previously represented first dimension of the matrix is changed to the previously represented first dimension of the matrix representing one sentence in a document. The CharCNN process is then operated with the maximum pooling of three layers on each row and the first dimension of the matrix produces a matrix with the dimensions (number of documents × 15 sentences) × 1 × 1 × 128. By adding two Highway Network layers, the matrix is transformed into the size of a document × 15 sentences × 128 [35]. 128 numbers now represent a single sentence.
2. On the BiGRU side, the input is transformed in the same way that it was in CharCNN, that is, if the first dimension of the matrix previously represents one document, now, it represents one sentence in a document. After that, the BiGRU process is carried out with the Hierarchical Attention Level Word to obtain a matrix of size (number of documents × 15 sentences) × 128. The matrix is transformed into the size of the number of documents times 15 sentences times 128. Now, one sentence has been represented by 128 numbers.

The outcomes of both sides of the method are merged or concatenated into a matrix with the dimensions: number of documents × 15 sentences × 256. The matrix from the word level is processed at the sentence level by the BiGRU process, Hierarchical Attention Level Sentence, and probability calculations for each sentence on the SumaRuNNer layer. The model's final form or output is a matrix with the size of the number of documents × 15 sentences, with one number representing the probability of

the sentence being included in the summarization in the range of values from 0 to 1. Five sentences with the highest probability are chosen as part of the summarization in this study.

C. Experiment Result

In this research, the HASumRuNNer model is tested with 7 types of gradient-based algorithms used during the backpropagation process of the model. The test is carried out with 5 times K-Fold validation where for each fold the amount of data tested is shown in Table II.

TABLE II. K-FOLD DATASET

Fold	Training	Testing
1	15019	3775
2	15019	3775
3	15019	3775
4	15020	3774

Each fold uses 10% of the total training data for data validation or development. The model is then trained using batch sizes of 128 and epochs of 3 times for the entire gradient-based algorithm, except Adadelta, which uses epochs of 18 times. The values of ROUGE-1, ROUGE-2, and ROUGE-L for each fold, as well as their averages, are displayed for each algorithm.

1) Stochastic gradient descent (SGD)

TABLE III. RESULT ROUGE-N FROM SGD

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
ROUGE-1	69.46	70.04	69.76	70.59	69.59	69.89
ROUGE-2	63.1	63.65	63.32	64.22	63.1	63.48
ROUGE-L	66.9	67.43	67.2	67.95	66.94	67.28

Based on Table III, the average ROUGE-1 value of the algorithm is 69.89. Then, the average ROUGE-2 value is 63.48 and the average ROUGE-L value is 67.28.

2) Adagrad

TABLE IV. RESULT ROUGE-N FROM ADAGRAD

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
ROUGE-1	69.45	69.95	69.61	70.29	69.45	69.75
ROUGE-2	63.09	63.55	63.16	63.92	62.95	63.33
ROUGE-L	66.9	67.35	67.05	67.66	66.8	67.15

Based on Table IV above, the average ROUGE-1 value of the algorithm is 69.75. Then, the average ROUGE-2 value is 63.33 and the average ROUGE-L value is 67.15.

3) RMSProp

TABLE V. RESULT FROM RMSPROP

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
ROUGE-1	70.11	71.06	70.78	70.9	70.36	70.64
ROUGE-2	63.7	64.71	64.4	64.56	63.88	64.25
ROUGE-L	67.54	68.48	68.27	68.42	67.82	68.11

Based on Table V, the average ROUGE-1 value of the algorithm is 70.64. Then, the average ROUGE-2 value is 64.25 and the average ROUGE-L value is 68.11.

4) Adam

TABLE VI. RESULT FROM ADAM

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
ROUGE-1	70.46	71.11	70.54	70.94	70.47	70.7
ROUGE-2	64.11	64.8	64.1	64.6	64.05	64.33
ROUGE-L	67.86	68.53	67.97	68.39	67.96	68.14

Based on Table VI, the average ROUGE-1 value of the algorithm is 70.7. Then, the average ROUGE-2 value is 64.33 and the average ROUGE-L value is 68.14.

5) Adadelta

TABLE VII. RESULT FROM ADADELTA

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
ROUGE-1	68.65	69.33	69.65	69.82	67.97	69.08
ROUGE-2	62.31	62.92	63.21	63.53	61.34	62.66
ROUGE-L	66.13	66.74	67.11	67.23	65.32	66.51

Based on Table VII, the average ROUGE-1 value of the algorithm is 69.08. Then, the average ROUGE-2 value is 62.66 and the average ROUGE-L value is 66.51.

6) Adamax

TABLE VIII. RESULT FROM ADAMAX

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
ROUGE-1	69.28	70.09	69.88	70.3	69.62	69.83
ROUGE-2	62.88	63.4	63.41	63.9	63.12	63.4
ROUGE-L	66.71	67.49	67.31	67.72	67.01	67.25

Based on Table VIII, the average ROUGE-1 value of the algorithm is 69.83. Then, the average ROUGE-2 value is 63.4 and the average ROUGE-L value is 67.25.

7) Nadam

TABLE IX. RESULT FROM NADAM

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
ROUGE-1	70.17	70.95	70.66	70.96	69.78	70.6
ROUGE-2	63.83	64.59	64.25	64.6	63.32	64.13
ROUGE-L	67.62	68.38	68.09	68.43	67.17	67.94

Based on Table IX, the average ROUGE-1 value of the algorithm is 70.6. Then, the average ROUGE-2 value is 64.13 and the average ROUGE-L value is 67.94.

D. Model Evaluation

Adam's algorithm produces the best ROUGE-1, ROUGE-2, and ROUGE-L values among the seven gradient-based algorithms used in the HASumRuNNer model, with the values of 70.7, 64.33, and 68.14. This may be due to Adam being a hybrid of the RMSProp and SGD with momentum methods, in which Adam uses the root of the gradient to increase the learning rate, as in RMSProp, and a moving average of the gradient, as in SGD with momentum. The difference in ROUGE-1, ROUGE-2, and ROUGE-L values generated by each algorithm also demonstrate that the use of different backpropagation methods can affect or be used as the parameters to improve

the model's performance, particularly deep learning-based models. Furthermore, for the comparison of the HASumRuNNer model with other methods used in research [36] can be seen in the Table X below.

TABLE X. COMPARISON HASUMRUNNER WITH OTHER MODEL

Model	ROUGE-1	ROUGE-2	ROUGE-L
HASumRuNNer + Adam	**70.7**	**64.33**	**68.14**
BAYES	62.70	54.32	61.93
HMM	17.62	4.70	15.89
MAXENT	50.94	44.33	50.26
NEURALSUM	67.60	61.16	66.86
NEURALSUM 300 emb. size	67.96	61.65	67.24
NEURALSUM + FASTTEXT	67.78	61.37	67.05

According to Table X, the proposed HASumRuNNer model has higher ROUGE-1, ROUGE-2, and ROUGE-L values than the other models mentioned. The HASumRuNNer model has a higher ROUGE-1 value of 2.74 when compared to the NEURALSUM 300 emb. size. The ROUGE-2 value is 2.68 times greater than the NEURALSUM 300 emb. size, and the ROUGE-L value is 1.09 times greater. By combining CharCNN and BiGRU at the word level, the HASumRuNNer model can produce a more accurate representation of words at the word level and sentence representations at the sentence level. The two methods allow the model to study word patterns in documents from two sides. Then, the hierarchical attention at the word-level and sentence-level also prevents the loss of information on every word in the document due to the length of the words or sentences that becomes the model input.

V. CONCLUSION AND FUTURE WORKS

This study contributes to the development of the HASumRuNNer model, which can provide a more accurate representation of words through the employment of two approaches, namely CharCNN and BiGRU, as well as the use of a hierarchical attention mechanism to minimize information loss in documents with long sentences or words. So, it supports users in texting extractive summarization with greater accuracy.

Adam, with values of 70.7, 64.33, and 68.14, is the gradient-based algorithm that produces the best ROUGE-1, ROUGE-2, and ROUGE-L values in the HASumRuNNer model. The proposed HASumRuNNer model is then appropriate and accurate enough to be used for extractive text summarization. The testing with the Indosum dataset demonstrates that the model, particularly the Adam gradient-based algorithm, produces higher ROUGE-1, ROUGE-2, and ROUGE-L values than the other models or methods used as references.

Testing the HASumRuNNer model with various datasets will increase the model's validity in the future work. Also, the summary of each document in the Indosum dataset utilized in this work is created in an abstract form, therefore the HASumRuNNer model is developed in an abstract method for text summarizing to produce a more accurate summary.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Research conceptualization was done by Muljono and Kevin Djajadinata. The research methodology was done by Mangatur Rudolf Nababan, Raden Arief Nugroho, and Muljono. Software preparation was done by Muljono and Kevin Djajadinata. Results validation was done by Mangatur Rudolf Nababan and Muljono. Formal analysis was analyzed by Kevin Djajadinata and Muljono. Research resources was prepared by Muljono. Investigation was worked by Kevin Djajadinata and Muljono. Data curation was worked by Kevin Djajadinata and Raden Arief Nugroho. The writing of the original draft preparation was done by Muljono and Raden Arief Nugroho. The review and editing were done by Mangatur Rudolf Nababan. The visualization was done by Kevin Djajadinata. Supervision was managed by Muljono and Raden Arief Nugroho. The project administration was managed by Kevin Djajadinata. All authors have read and approved the final manuscript.

FUNDING

This article is funded by the Ministry under Hibah Penelitian Dasar (Fundamental Research Grant) 2022.

ACKNOWLEDGEMENT

Regarding data collection, the authors appreciate the help provided by Kevin Djajadinata, an alumni of Informatics Engineering Department of Universitas Dian Nuswantoro. Furthermore, the authors would also like to thank Universitas Sebelas Maret and Universitas Dian Nuswantoro for the continuous support in completing this study. The authors of this study express sincere gratitude to the Ministry of Education, Culture, Research, and Technology.

REFERENCES

[1] E. R. Mahalleh and F. S. Gharehchopogh, "An automatic text summarization based on valuable sentences selection," *International Journal of Information Technology*, vol. 14, no. 6, pp. 2963–2969, Oct. 2022, doi: 10.1007/S41870-022-01049-X/TABLES/3

[2] R. Adelia, S. Suyanto, and U. N. Wisesty, "Indonesian abstractive text summarization using bidirectional gated recurrent unit," *Procedia Comput. Sci.*, vol. 157, pp. 581–588, Jan. 2019, doi: 10.1016/J.PROCS.2019.09.017

[3] N. Alami, M. Meknassi, and N. En-nahnahi, "Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning," *Expert Syst. Appl.*, vol. 123, pp. 195–211, Jun. 2019, doi: 10.1016/J.ESWA.2019.01.037

[4] D. Anand and R. Wagh, "Effective deep learning approaches for summarization of legal texts," *Journal of King Saud University—Computer and Information Sciences*, vol. 34, no. 5, pp. 2141–2150, May 2022, doi: 10.1016/J.JKSUCI.2019.11.015

[5] V. Andrearczyk and P. F. Whelan, "Deep learning in texture analysis and its application to tissue image classification,"

Biomedical Texture Analysis: Fundamentals, Tools and Challenges, pp. 95–129, Jan. 2017, doi: 10.1016/B978-0-12-812133-7.00004-1

[6] D. Bacciu and A. Bruno, "Text summarization as tree transduction by top-down TreeLSTM," in *Proc. the 2018 IEEE Symposium Series on Computational Intelligence*, pp. 1411–1418, Jan. 2019, doi: 10.1109/SSCI.2018.8628873

[7] R. Bhargava, G. Sharma, and Y. Sharma, "Deep text summarization using generative adversarial networks in Indian languages," *Procedia Comput. Sci.*, vol. 167, pp. 147–153, Jan. 2020, doi: 10.1016/J.PROCS.2020.03.192

[8] R. Bhargava and Y. Sharma, "Deep extractive text summarization," *Procedia Comput. Sci.*, vol. 167, pp. 138–146, Jan. 2020, doi: 10.1016/J.PROCS.2020.03.191

[9] R. Chandraseta and M. L. Khodra, "Composing Indonesian paragraph for biography domain using extractive summarization," in *Proc. 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, ICAICTA 2019*, Sep. 2019, doi: 10.1109/ICAICTA.2019.8904118

[10] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," in *Proc. 54th Annual Meeting of the Association for Computational Linguistics*, Mar. 2016, vol. 1, pp. 484–494, doi: 10.18653/v1/p16-1046

[11] Y. Diao, *et al.*, "CRHASum: Extractive text summarization with contextualized-representation hierarchical-attention summarization network," *Neural Comput. Appl.*, vol. 32, no. 15, pp. 11491–11503, Aug. 2020, doi: 10.1007/S00521-019-04638-3/FIGURES/3

[12] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks," in *Proc. the International Conference on Computational Techniques, Electronics and Mechanical Systems, CTEMS 2018*, 2018, pp. 92–99, doi: 10.1109/CTEMS.2018.8769211

[13] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for (parallel) stochastic optimization," in *Proc. the IEEE Conference on Decision and Control*, 2012, pp. 5442–5444, doi: 10.1109/CDC.2012.6426698

[14] C. Feng, H. Chen, F. Cai, and M. Rijke, "Attentive encoder-based extractive text summarization," in *Proc. the International Conference on Information and Knowledge Management*, Oct. 2018, pp. 1499–1502, doi: 10.1145/3269206.3269251

[15] A. Shaddeli, F. Soleimanian Gharehchopogh, M. Masdari, and V. Solouk, "An improved African vulture optimization algorithm for feature selection problems and its application of sentiment analysis on movie reviews," *Big Data and Cognitive Computing*, vol. 6, no. 4, p. 104, Sep. 2022, doi: 10.3390/BDCC6040104

[16] A. Hosseinalipour, F. S. Gharehchopogh, M. Masdari, and A. Khademi, "A novel binary farmland fertility algorithm for feature selection in analysis of the text psychology," *Applied Intelligence*, vol. 51, no. 7, pp. 4824–4859, Jul. 2021, doi: 10.1007/S10489-020-02038-Y/FIGURES/28

[17] A. Shaddeli, F. S. Gharehchopogh, M. Masdari, and V. Solouk, "BFRA: A new binary hyper-heuristics feature ranks algorithm for feature selection in high-dimensional classification data," *International Journal of Information Technology & Decision Making (IJITDM)*, vol. 22, no. 01, pp. 471–536, Jan. 2023, doi: 10.1142/S0219622022500432

[18] H. K. Maragheh, F. S. Gharehchopogh, K. Majidzadeh, and A. B. Sangar, "A new hybrid based on long short-term memory network with spotted hyena optimization algorithm for multi-label text classification," *Mathematics*, vol. 10, no. 3, p. 488, Feb. 2022, doi: 10.3390/MATH10030488

[19] W. Guo, B. Wu, B. Wang, and Y. Yang, "Two-stage encoding extractive summarization," in *Proc. the 2020 IEEE 5th International Conference on Data Science in Cyberspace, DSC 2020*, 2020, pp. 346–350, doi: 10.1109/DSC50466.2020.00060

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735

[21] Y. Chen. (Aug. 2015). Convolutional neural network for sentence classification. [Online]. Available: https://uwspace.uwaterloo.ca/handle/10012/9592

[22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. the 3rd International Conference for Learning Representations*, 2015.

[23] B. Kitchenham and S. M. Charters. (2007). Guidelines for performing systematic literature reviews in software engineering. [Online]. Available: https://www.researchgate.net/publication/302924724

[24] K. Kurniawan and S. Louvan, "IndoSum: A new benchmark dataset for Indonesian text summarization," in *Proc. the 2018 International Conference on Asian Language Processing, IALP 2018*, 2019, pp. 215–220, doi: 10.1109/IALP.2018.8629109

[25] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *Proc. the 2019 Conference on Empirical Methods in Natural Language Processing*, Aug. 2019, pp. 3730–3740, doi: 10.18653/v1/d19-1387

[26] K. Lv, S. Jiang, and J. Li, "Learning gradient descent: Better generalization and longer horizons," in *Proc. 34th International Conference on Machine Learning*, 2017, pp. 2247–2255.

[27] R. Nallapati, F. Zhai, and B. Zhou, "SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents," in *Proc. the AAAI Conference on Artificial Intelligence*, Feb. 2017, vol. 31, no. 1, pp. 3075–3081, doi: 10.1609/AAAI.V31I1.10958

[28] C. Shah and A. Jivani, "A hybrid approach of text summarization using latent semantic analysis and deep learning," in *Proc. 2018 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2018*, Nov. 2018, pp. 2039–2044, doi: 10.1109/ICACCI.2018.8554848

[29] A. Sinha, A. Yadav, and A. Gahlot, "Extractive text summarization using neural networks," arXiv preprint, arXiv:1802.1013, https://arxiv.org/abs/1802.10137v1

[30] A. Hosseinalipour, F. S. Gharehchopogh, M. Masdari, and A. Khademi, "Toward text psychology analysis using social spider optimization algorithm," *Concurr. Comput.*, vol. 33, no. 17, p. e6325, Sep. 2021, doi: 10.1002/CPE.6325

[31] R. K. Srivastava, K. Greff, K. Ch, and J. U. Schmidhuber, "Highway networks," arXiv preprint, arXiv:1505.00387, https://arxiv.org/abs/1505.00387v2

[32] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020, doi: 10.1109/TCYB.2019.2950779

[33] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. the 30th International Conference on Machine Learning*, 2013, pp. 1139–1147.

[34] V. Kieuvongngam, B. Tan, and Y. Niu, "Automatic text summarization of COVID-19 medical research articles using BERT and GPT-2," arXiv preprint, arXiv:2006.01997, Jun. 2020, https://arxiv.org/abs/2006.01997v1

[35] P. Yan, L. Li, and D. Zeng, "A shortcut-stacked document encoder for extractive text summarization," in *Proc. the International Joint Conference on Neural Networks*, Jul. 2019, doi: 10.1109/IJCNN.2019.8852051

[36] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," arXiv preprint, arXiv:1212.5701, Dec. 2012, https://arxiv.org/abs/1212.5701v1