Extractive text summarization for scientific journal articles using long short-term memory and gated recurrent units

Devi Fitrianah, Raihan Nugroho Jauhari
Department of Informatics, Faculty of Computer Science Universitas Mercu Buana, Jakarta, Indonesia

Article Info

Article history:

Received Jun 12, 2021
Revised Sep 27, 2021
Accepted Dec 29, 2021

Keywords:

Extractive approach
Gated recurrent units
Journal articles
Long short-term memory
Text summarization

ABSTRACT

Along with the increasing number of scientific publications, many scientific communities must read the entire text to get the essence of information from a journal article. This will be quite inconvenient if the scientific journal article is quite long and there are more than one journals. Motivated by this problem, encourages the need for a method of text summarization that can automatically, concisely, and accurately summarize a scientific article document. The purpose of this research is to create an extractive text summarization by doing feature engineering to extract the semantic information from the original text. Comparing the long short-term memory algorithm and gated recurrent units and were used to get the most relevant sentences to be served as a summary. The results showed that both algorithms yielded relatively similar accuracy results, with gated recurrent units at 98.40% and long short-term memory at 98.68%. The evaluation method with matrix recall-oriented understudy for gisting evaluation (ROUGE) is used to evaluate the summary results. The summary results produced by the LSTM model compared to the summary results using the latent semantic analysis (LSA) method were then obtained recall values at ROUGE-1, ROUGE-2, and ROUGE-L respectively were 76.25%, 59.49%, and 72.72%.

Corresponding Author:

Raihan Nugroho Jauhari
Faculty of Computer Science, Universitas Mercu Buana
Jalan Raya Meruya Selatan no. 1, Kembangan Jakarta Barat-16550, Indonesia
Email: raihanrnj@gmail.com

1. INTRODUCTION

A journal article is a scientific publication containing academic articles written for the scientific community. In general, journal articles are also accessible to the public. But the target audience is narrow including academics, students, researchers, and other student communities. Most readers often want a summary of a journal article quickly without reading the contents of the entire scientific article document. However, it is difficult for people to manually extract summaries from large text documents [1]. A text summarization system is thus required to help automatically obtain summary results from a document. A summary is the result of a collection of words, sentences, and paragraphs that are less than half the length of the original text [2]-[4]. With this summarization system, it is expected that readers will acquire summaries containing important sentences easily, quickly, and precisely without having to read the entire content of the original source text.

There are two types of text compactions that are most popular. First, extractive techniques extract important sentences from document texts and combine them into a summary [5], [6]. Usually, the resulting sentences are arranged the same as the sentences in the original document. The second type is the abstractive technique, which produces summary sentences that are different from the original sentence since humans take the essence of a document read. Abstract summaries start with reducing the main concept of the document in fewer words [7]. Thus, the resulting sentences may seem simpler, natural, and not rigid [8]. Various studies in the field of automated text summarization using extractive methods have been widely done. Previous research was conducted by Prabowo *et al.* [9], where documents were summarized by filtering important words using sentence scoring with the term frequenc -inverse document frequency (TF-IDF) method. In the TF-IDF method, the value of each word is calculated based on the frequency of its occurrence in the document [10]. The sentence-scoring process is obtained by summarizing the weight of each word so that the final weight of each sentence is produced [11]. Summary results were obtained from the sentences that had the highest rank at the time of the weighting process. This study successfully produced summary results related to the main topic of the document text. However, optimization is still needed in the selection of sentences that will be used as a summary result not only by calculating the sentences that have the highest weight, one solution is to extract hidden features from the text. Generally, the sentence with the highest weight is a sentence containing important or certain scientific words so that the sentence is less descriptive to be summarized [12], [13].

Geetha [14] conducted a study in the case of Kannada language. Geetha uses the latent semantic analysis (LSA) method, which extracts semantic structures or hidden meanings in a sentence and then produces a general or broad meaningful summary. The LSA method applies a linear singular value decomposition (SVD) algebraic approach by forming a representation matrix based on term associations, which are words in related documents based on TF-IDF weighting [15]. However, one of the shortcomings of the LSA is that the representation obtained is not explicit and can degrade the performance process for large and multilingual documents. Still, the LSA has the advantage that the summary results consistently represent the conceptual relationship between words, sentences, and paragraphs [16]. Concerning the studies mentioned above, this study proposes extractive text summarization that can perform text summarization against a scientific journal article that has a varied structure and tagging. In order to extract likelier semantic meanings that are contained in a document, the researchers used weighted sentences by doing feature engineering, which includes sentence-length cut-off features, fixed phrase features, paragraph location, thematic word features, and uppercase word features. Then from these features, the classification process was done by using the long short-term memory (LSTM) algorithm and gated recurrent units (GRU) to determine the most relevant sentences to the topic and represent the entire text.

Simple neural networks are not sufficient for handling classification cases against sequential data, such as text data consisting of strings of words and sentences [17]. The LSTM network is a variant of the recurrent neural network (RNN). LSTM is popular because it can reduce the problem of explosion and diffusion gradients [18]. LSTM transforms the memory structure of cells in the RNN by transforming the *tanh* activation function layer in the RNN into a structure containing memory units and gate mechanisms. It aims to decide how to utilize and update information stored in memory cells. Due to this structure, this reduces the problem of gradient diffusion glue and explosion [19], [20]. LSTM is proposed to overcome the occurrence of vanishing gradients in RNN when processing long sequential data. RNN has several architectures, one of which is the GRU. The GRU has a "gate" reminder to bring information from previous circumstances. GRU has also undergone many developments, one of which is bidirectional GRU (BiGRU), which applies the concept of GRU in a forward and backward manner so that it has information from the previous and the following states [21] and so that it knows the surrounding information.

2. RESEARCH METHOD

The method in this research will take several stages. The first is the process of collecting datasets of scientific journal articles obtained from open libraries on the internet. Then the data will be extracted with metadata such as the abstract section, title, author, and year of publication. After that, the dataset will go through the pre-processing stage and then proceed to the feature engineering stage to extract semantic information from the text. After the features have been extracted, 75 final features will be generated, all of which are binary values. In labeling the data, the researcher used the latent semantic analysis technique to find the most important sentence in the text. At the model training stage, the researchers compared two algorithms: LSTM and GRU. Finally, the model evaluation is used to determine how well the model performs and which algorithm gives the best results and performance. Figure 1 describes the stages in this research.

2.1. Dataset collection

Dataset collection is done manually by downloading scientific articles from the Arxiv open access repository. Scientific journal article data, as many as 41,000 research topics related to artificial intelligence, machine learning, and computer vision, were taken from the open-source library. The PDF-formatted set of articles will then go through the dataset-build stage to extract the metadata and proceed to the data-labeling process.

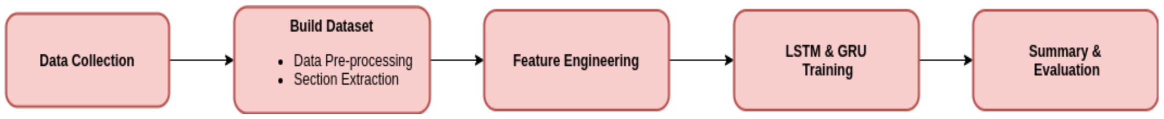

Figure 1. Research methodology

2.2. Build datasets

At this stage, 41,000 scientific journal articles in PDF format that have previously been collected will go through pre-processing data and meta-data extraction to be used as attributes in the dataset.

2.2.1. Pre-processing

The pre-processing stage is related to preparing a number of data results for the dataset build, which will then be used for learning materials for computers. These data are some of the components that determine how well the model's results are built. In this step, we conduct some data-cleaning processes, including the following: i) uniform words by changing all text letters to lowercase and removing some character letters that are regarded as delimiters; this process is called case folding [22]; ii) the next step is the tokenization process by separating the word string from the sentencing document; iii) stopword is a process used to eliminate words that are considered unimportant [23]; iv) the final step of preprocessing stems from changing the form of the word to make it the root word.

2.2.2. Create the dataset

Section extraction is done automatically using Python to generate attributes, such as the title, abstract, and year. After obtaining the attribute, the next step is the data-labeling process. In this study, the researchers tried to use abstract attributes as original texts for compaction. Target data or labels will be generated automatically using the LSA method to generate summary results.

The summary result is then encoded into an index sentence that represents the sentence index in the text. The sentence index is stored in the summary column and will be used as the target column in the dataset. After the data-labeling process, the dataset will have 41,000 rows of data with several columns, including title, abstract, year, and summary. The stages in building the dataset are illustrated in Figure 2.

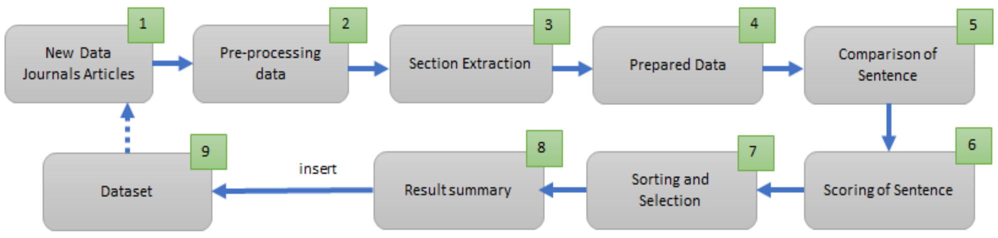

Figure 2. Stages of building the dataset

2.3. Feature engineering

At this stage, feature engineering extracts new features that will later be used for scoring each sentence. Feature engineering uses abstract columns to make or organize several categories of feature scoring, which are as shown in:

2.3.1. Sentences-length cut-off features

Usually, short sentences tend not to be included in summaries [24]. Here the threshold is given for each sentence (e.g., 20 words). If the sentence contains more words than the threshold length, then this feature will be worth 1, and if it contains fewer words than the threshold length, then the feature will be worth 0.

$$f(x) = \left\{ \frac{1\ if\ sentence\_lenght > threshold}{0\ otherwhise} \right.$$

2.3.2. Fixed phrase features

Some phrases including the conclusion, summary, and essence tend to be representative of summaries and generally imply that sentences containing those key phrases are important sentences. There are 26 indicator phrases; if a sentence contains at least one of these keywords, then the feature will be worth 1 and if it does not contain these keywords, then the feature will be worth 0.

$$f(x)=\{ \frac{1\ if\ frase\_indicator\ in\ sentence}{0\ otherwhise}$$

2.3.3. Paragraph location

The position of a sentence in a paragraph in general also tends to have in identifying whether a sentence is repressive to the summary or not [25]. This feature is discrete (3, 2, 1), so the sentence will be categorized into three groups: the sentence at the beginning of the sentence is 3, the sentence in the middle of the paragraph is 2, and the sentence at the end of the paragraph is 1.

$$f(x)=\{ \frac{3\ if\ sentence\ at\ the\ beginning\ of\ the\ paragraph}{2\ if\ sentence\ at\ the\ middle\ of\ the\ paragraph\ ,and\ 1\ otherwhise}$$

2.3.4. Thematic word features

The two or three selected words that appear most frequently in the text (excluding stopwords) are called thematic words. Sentences containing at least one thematic word will be 1, and if there is no thematic word in it, it will be 0.

$$f(x)=\left\{ \frac{1\ if\ thematic\_word\ in\ sentence}{0\ otherwhise} \right.$$

2.3.5. Uppercase word features

In general, words that contain capital letters, such as the example acronym ASA (American Statistical Association), tend to represent the importance of the sentence's in a paragraph. This feature will be binary; sentences are sorted by the frequency of words with the most capital letters. The top-ranked sentence will receive a value of 1; otherwise, it will be 0.

$$f(x)=\{ \frac{1\ if\ sentence=top\_uppercase\_sentence}{0\ otherwhise}$$

2.4. Model training

The ready data will be divided into two parts: the training set and the test set. The training set data will be used during the training process, while the test set data will be used to perform performance tests on the model. There are 75 features resulting from the feature engineering process that will be used as inputs. This input will then be processed by LSTM or GRU cells to produce an output that can then be channeled to the hidden layer. In hidden layers, the trained data will be stored in the logistical form to continuously update the weight and other values to meet the target output. To obtain weights that are close to the output value, the decoder uses the attention layer to focus the result of the previously hidden layer. After that, the process of decoding will be done to the sentence index data to form the text, which will then be displayed as the output result.

2.5. Evaluation

The evaluation stage calculates accuracy by obtaining precision, recall, and F-Measure values. Precision is a measurement of the accuracy of the summary results produced by automatic text summarization, and it can measure how relevant the chances are of a text being taken to be used as a summary. Recall is a measurement of the summary success value generated by the system and can measure a relevant text's chances of being taken as a summary result. F-measure is the value used to determine the accuracy level of the summary results obtained.

$$\text{Accuracy}=\frac{TP+TN}{TP+TN+FP+FN} \quad \text{Precision}=\frac{TP}{TP+FP} \quad \text{Recall}=\frac{TP}{TP+FN} \quad \text{F1 Score}=\frac{Precision\ x\ Recall}{Precision + Recall}$$

where: TP=true positive is a sentence that is in the summary of the LSA and appears in the summary of the system
FP=false positive sentence that is in the summary of the LSA but does not appear in the system
FN=false negative is a sentence that is in the summary of the LSA but does not appear in the system
TP=true negative is a sentence that is not in the LSA summary or system summary

3. RESULTS AND DISCUSSION

3.1. Train-test split evaluation

This study produced the value of accuracy for the summary results by predicting the implementation of classification methods. The models used for testing were the LSTM comparison algorithm and gated recurrent units. Table 1 shows the results of split test data sharing, with the best results being 60% train set data and 40% test set data with a value of 98.7%.

Table 1. The accuracy result is based on the distribution of the train test split

Split train test	Accuracy LSTM	Accuracy GRU
90/	0.9849	0.9822
80/20	0.9857	0.9829
70/30	0.9853	0.9856
60/40	**0.9871**	0.9859

3.2. Hyperparameters tuning

In this research scenario, we performed several hyperparameter settings to influence the output results and improve model performance. The hyperparameters used included epoch, batch_size, learning rate, and neuron activation function.

a. Epoch and batch size
The results of batch size and epoch adjustment are shown in Table 2. The result provides an LSTM accuracy of 93.73% with a total batch_size of 32 and epoch 100, while the compute time is 386.7663 seconds.

b. Learning rate
The results of the tuning learning rate are shown in Table 3. The best accuracy is LSTM with a value of 0.9860 when the learning rate is 0.001. The computing time in this process was 386.8756 seconds.

c. Algorithm optimization
The scenario results for the tuning optimizer algorithm are shown in Table 4. This experiment demonstrated the best accuracy with LSTM at 98.60% when using the Adam optimizer algorithm. The duration of the process in this scenario is 386.7442 seconds.

d. Neuron activation function
Table 5 displays the results of the tuning activation function experiment. The best result accuracy was achieved by LSTM, with a value of 98.60% when using the hard_sigmoid function. The computed time to process this experiment was 386.9453 seconds.

Table 2. The accuracy score of batch size and epoch tuning

Algoritma	Epoch/Batch_size	Epoch=30	Epoch=50	Epoch=100
GRU	Bs=20	0.9857	0.9835	0.9852
	Bs=32	0.9849	0.9868	0.9840
	Bs=40	0.9863	0.9851	0.9851
LSTM	Bs=20	0.9855	0.9868	0.9872
	Bs=32	0.9860	0.9870	**0.9873**
	Bs=40	0.9871	0.9870	0.9869

Table 3. The accuracy score of learning rate tuning

Learning rate	Accuracy LSTM	Accuracy GRU
0.001	**0.9860**	0.9802
0.01	0.9486	0.9758
0.1	0.9486	0.9752

Table 4. The accuracy score of optimizer algorithm tuning

Optimizer	Accuracy LSTM	Accuracy GRU
SGD	0.9542	0.9745
RMSProp	0.9803	0.9852
Adagrad	0.9696	0.6683
Adadelta	0.9544	0.9739
Adam	**0.9860**	0.9802

Table 5. The accuracy score of activation function tuning

Activation function	Accuracy LSTM	Accuracy GRU
Softmax	0.9865	0.9857
Relu	0.9822	0.9853
tanh	0.9583	0.9802
Sigmoid	0.9838	0.9864
Hard Sigmoid	**0.9873**	0.9859

3.3. Model testing and evaluation

At this stage, we evaluated the LSTM and GRU models against a dataset of scientific journal articles that we had previously conducted in the training process. Table 6 shows a comparison of the evaluation values between LSTM and GRU. It can be seen that LSTM gave relatively similar results compared to GRU, with a GRU accuracy of 98.40% and an LSTM of 98.68%. At this stage, we also applied all parameter value settings that had the best accuracy composition from previous experiments, including batch_size=32, epoch=100, learning_rate=0.001, optimizer=Adam, and neuron activation function=hard_sigmoid to our model. Table 7 shows the evaluation values for LSTM before and after setting the parameters. The results showed that after tuning the parameters, the accuracy score increased from 98.33% to 98.68%. We defined every 15 sentences in the text with a label to help us choose which label to represent as a summary. The accuracy results of the 15 labels are shown in Figure 3.

Table 6. Evaluation report between LSTM and GRU

Evaluation Score	LSTM	GRU
F1-Score	0.9826	0.9750
Precision	0.9756	0.9635
Recall	0.9906	0.9879
Accuracy	0.9868	0.9840
Time (seconds)	386	303 (21.50% faster)

Table 7. LSTM evaluation report before and after parameter tunning

Evaluation Score	LSTM before tunning	LSTM after tunning
F1-Score	0.9368	0.9826
Precision	0.9623	0.9756
Recall	0.9279	0.9906
Accuracy	0.9833	**0.9868**
Time (seconds)	386	386

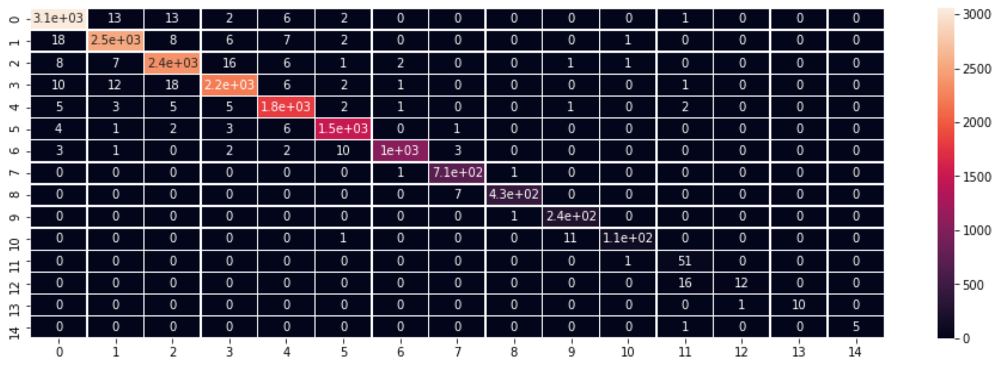

Figure 3. Confusion matrix results

3.4. Test summarizing the text

After the LSTM model was determined as the best model, and the most optimal hyper tuning parameter settings were selected. Then we tested the model to summarize the abstract section of journal articles manually inputted into the system compared with the summary results using the LSA. Table 8 shows displays the results of this comparison.

Table 8. Comparison of LSA summary results and LSTM model summaries

Text original	Summary generate by LSA	Summary generate by LSTM model
The detection of Covid-19 is an important step to make early for suspected Covid-19 patients can be done further steps. One way of detection is through x-ray images of the lungs. However, besides that, we need an algorithm model that can produce high language, light computation What is needed so that it can be applied in a detection device. Models in CNN can performs detection with accuracy but tends to require large memory usage. CNN with fewer parameters can provide both storage and memory usage so that it can be processed in real time both in the form of a detection tool and decision making via the cloud. Besides, CNN with smaller parameters it can also be applied to FPGA and other hardware has limited memory capacity. To produce accurate detection of COVID-19 on lung x-ray images but the computation is also light, we CNN architecture is small but reliable to use a channel exchange technique called ShuffleNet. In this study, we tested and compared capabilities of ShuffleNet, EfficientNet, and ResNet50 because they have a smaller number of parameters compared to CNN in general such as VGGNet or FullConv which uses a convolutional layer full but has capable detection capabilities. We used 1125 x-ray images and achieved an accuracy of 86.93% with the number of model parameters that were 18.55 times less than EfficientNet and 22.36 times less than ResNet50 to wait for 3 categories namely Covid-19, Pneumonia, and normal through a 5-fold cross validation test. The memory required by each of these CNN architectures for performs detection linearly with the number of parameters where ShuffleNet only requires GPU memory of 0.646 GB or 0.43 times that of ResNet50, 0.2 times of EfficientNet, and 0.53 times times from FullConv. Furthermore, ShuffleNet performs the fastest detection of 0.0027 seconds.	To produce accurate detection of COVID-19 on lung x-ray images but the computation is also light, we CNN architecture is small but reliable to use a channel exchange technique called ShuffleNet. We used 1125 x-ray images and achieved an accuracy of 86.93% with the number of model parameters that were 18.55 times less than EfficientNet and 22.36 times less than ResNet50 to wait for 3 categories namely Covid-19, Pneumonia, and normal through a 5-fold cross validation test.	to produce accurate detection of covid 19 on lung x ray images but the computation is also light we cnn architecture is small but reliable to use a channel exchange technique called shufflenet we used 1125 x ray images and achieved an accuracy of 8693 with the number of model parameters that were 1855 times less than efficientnet and 2236 times less than resnet50 to wait for 3 categories namely covid 19 pneumonia and normal through a 5 fold cross validation test

To determine the summary accuracy results generated by the LSTM and LSA models from the table above, we used ROUGE. Table 9 shows that ROUGE-1, which measures the match rate of unigrams, gets a precision score of 75% between the LSTM model compared to the LSA results; ROUGE-2, which measures the match rate of a bigram, gets a precision score of 59%; and ROUGE-L, which measures the longest common subsequence (LCS), gets a precision score of 74%.

Table 9. Evaluation results of the ROUGE metric against the summary of the model and summary of the LSA

Evaluation Score	F1 Score	Precision	Recall
ROUGE-1	0.7577	0.7530	0.7625
ROUGE-2	0.5911	0.5875	0.5949
ROUGE-L	0.7441	0.7619	0.7272

4. CONCLUSION

From all the experimental scenarios conducted in this study, the results show that the new method that we introduced by combining one-hot encoding for each sentence and feature engineering to extract semantic meaning can produce extractive summaries that are just as effective as those done by the LSA algorithm. Different from LSA, because we use a deep-learning approach, our method can also be customized by changing the data-labeling method using another approach such as clustering or manual methods. GRU and LSTM provide very similar accuracy performances of 98.40% and 98.68%, respectively. LSTM performed slightly better because it has a more complex gate architecture, while GRU excels in the time it takes in the training process because each cell has only two gates. Tuning parameters also improved accuracy from 98.33% to 98.68% for LSTM. The configuration values on tuning parameters that produce the best performance are batch_size=32, epoch=100, learning_rate=0.001, optimizer=Adam, and neuron activation function=hard_sigmoid.

REFERENCES

[1] A. Quadros, I. No and J. Pinto, “Text Summarizer for URL and .DOCX files,” *International Journal of Advanced Research in Computer Science*, vol. 11, no. 4, pp. 18-22, 2020, doi: 10.26483/ijarcs.v11i4.6639.

[2] T. Uçkan and A. Karcı, “Extractive multi-document text summarization based on graph independent sets,” *Egyptian Informatics Journal*, vol. 21, pp. 145-157, 2020, doi: 10.1016/j.eij.2019.12.002.

[3] A. Pramita *et al.*, “Review of automatic text summarization techniques & methods,” *Journal of King Saud University - Computer and Information Sciences*, 2020, doi: 10.1016/j.jksuci.2020.05.006.

[4] A. K. Singh and M. Shashi, “Deep Learning Architecture for Multi-Document Summarization as a cascade of Abstractive and Extractive Summarization approaches,” *International Journal of Computer Sciences and Engineering*, vol. 7, no. 3, pp. 950-954, 2019, doi: 10.26438/ijcse/v7i3.950954.

[5] W. Yulita, S. Priyanta, and A. SN, “Automatic Text Summarization Based on Semantic Networks and Corpus Statistics,” *JCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 13, no. 2, p. 137, 2019, doi: 10.22146/ijccs.38261.

[6] Chu, Edward & Huang, Zi-Zhe, “DBOS: A Dialog-Based Object Query System for Hospital Nurses,” *Sensors*, vol. 2, pages. 6639, 2016, doi: 10.3390/s20226639.

[7] S. M. Patel, V. Dabho and H. B. Prajapati, “Extractive Based Automatic Text Summarization,” *Journal of Computers*, vol. 12, no. 6, pp. 550-563, 2017, doi: 10.17706/jcp.12.6.550-563.

[8] N. Bansal, A. Sharma and R. K. Singh, “Recurrent neural network for abstractive summarization of documents,” *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, p. 65-72, 2020, doi: 10.1080/09720529.2020.1721873.

[9] D. A. Prabowo, M. Fadli, M. A. Najib, H. A. Fauzi, and I. Cholissodin, “TF-IDF-Enhanced Genetic Algorithm Untuk Extractive Automatic Text Summarization,” *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 3, p. 208, 2016, doi: 10.25126/jtiik.201633217.

[10] Á. Hernández-Castañeda, R. A. García-Hernández, Y. Ledeneva and C. E. Millán-Hernández, "Extractive Automatic Text Summarization Based on Lexical-Semantic Keywords," in *IEEE Access*, vol. 8, pp. 49896-49907, 2020, doi: 10.1109/ACCESS.2020.2980226.

[11] S. Lagrini, M. Redjimi and N. Azizi, “Automatic Arabic Text Summarization Approaches,” *International Journal of Computer Applications*, vol. 164, no. 5, pp. 31-37, 2017, doi: 10.5120/ijca2017913628.

[12] Y. Du and H. Huo, "News Text Summarization Based on Multi-Feature and Fuzzy Logic," in *IEEE Access*, vol. 8, pp. 140261-140272, 2020, doi: 10.1109/ACCESS.2020.3007763.

[13] G. Shang, W. Ding, Z. Zhang, A. J.-P. Tixer, P. Meladianos, M. Vazirgiannis and J. Lorre, “Unsupervised Abstractive Meeting Summarization with Multi-Sentence Compression and Budgeted Submodular Maximization,” *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2018, doi: 10.18653/v1/P18-1062, 2018.

[14] J. K. Geetha and N. Deepamala, "Kannada text summarization using Latent Semantic Analysis," *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 1508-1512, doi: 10.1109/ICACCI.2015.7275826.

[15] D. Cai, L. Chang and D. Ji, "Latent semantic analysis based on space integration," *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, 2012, pp. 1430-1434, doi: 10.1109/CCIS.2012.6664621.

[16] P. Kherwa and P. Bansal, "Latent Semantic Analysis: An Approach to Understand Semantic of Text," *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 2017, pp. 870-874, doi:

10.1109/CTCEEC.2017.8455018.
[17] R. Nallapati, F. Zhai and B. Zhou, "Summarunner: a recurrent neural network-based sequence model for extractive summarization of documents," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 3075-3081, 2016, doi: 10.5555/3298483.3298681.
[18] R. Khan, Y. Qian and S. Naeem, "Extractive based Text Summarization Using KMeans and TF-IDF," vol. 11, no. 3, pp. 33-44, 2019, doi: 10.5815/ijieeb.2019.03.05.
[19] W. Huang, G. Rao, Z. Feng and Q. Cong, "LSTM with sentence representations for Document-level Sentiment Classification," *Neurocomputing*, vol. 308, no. 45, pp. 49-57, 2018, doi: 10.1016/j.neucom.2018.04.045.
[20] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, Doha, Qatar, pp. 1724-1734, 2014, doi: 10.3115/v1/D14-1179.
[21] R. Adelia, S. Suyanto and U. Wisesty, "Indonesian Abstractive Text Summarization Using Bidirectional Gated Recurrent Unit," *Procedia Computer Science*, vol. 157, pp. 581-588, 2019, doi: 10.1016/j.procs.2019.09.017.
[22] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, A. Mehmood and M. T. Sadiq, "Document-Level Text Classification Using Single-Layer Multisize Filters Convolutional Neural Network," *in IEEE Access*, vol. 8, pp. 42689-42707, 2020, doi: 10.1109/ACCESS.2020.2976744.
[23] K. Shetty and J. S. Kallimani, "Automatic extractive text summarization using K-means clustering," *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2017, pp. 1-9, doi: 10.1109/ICEECCOT.2017.8284627.
[24] M. -H. Su, C. -H. Wu and H. -T. Cheng, "A Two-Stage Transformer-Based Approach for Variable-Length Abstractive Summarization," *in IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2061-2072, 2020, doi: 10.1109/TASLP.2020.3006731.
[25] Y. Chen and Q. Song, "News Text Summarization Method based on BART-TextRank Model," *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2021, pp. 2005-2010, doi: 10.1109/IAEAC50856.2021.9390683.

BIOGRAPHIES OF AUTHORS

Devi Fitrianah she received the Bachelor's degree in Computer Science from Bina Nusantara University, Jakarta, Indonesia, in 2000, and the Master's degree in Information Technology and Ph.D. degree in Computer Science from the Universitas Indonesia, Depok, Indonesia, in 2008 and 2015, respectively. In 2014, she had a sandwich program at the Laboratory for Pattern Recognition and Image Processing and GIS (PRIPGIS Lab) Department of Computer Science, Michigan State University, East Lansing, Michigan, USA. She is currently a Faculty Member with the Department of Computer Science, Universitas Mercu Buana. Her research interests include machine learning, data mining, applied remote sensing, and geographic information system. She can be contacted at email: devi.fitrianah@mercubuana.ac.id.

Raihan Nugroho Jauhari received his bachelor's degree in Computer Science from Universitas Mercu Buana, Jakarta Indonesia in 2021. His concentration is on data science specialization and his research interest is in machine learning. He can be contacted at email: raihanrnj@gmail.com.