Computer Science and Information Technologies
Vol. 4, No. 2, July 2023, pp. 119~126
ISSN: 2722-3221, DOI: 10.11591/csit.v4i2.p119-126

The observed preprocessing strategies for doing automatic text summarizing

Muhammad Farhan Juna, Mardhiya Hayaty
Department of Informatics, Faculty of Computer Science, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia

Article Info

Article history:

Received Jan 10, 2023
Revised May 11, 2023
Accepted May 29, 2023

Keywords:

Automatic text summarization
Data cleaning
Rouge
Text preprocessing
Text summarizes

ABSTRACT

It is challenging for humans to keep up with the rapid creation of digital information due to the explosion of digital information. A written document can be analyzed to extract meaningful information using automatic text summarization. This research proposes 16 different experimental settings in which the model developed by IndoBERT will be applied in order to answer the question of how much of an impact preprocessing has on the quality of summaries produced by automatic text summarization. In order to answer this question, the researchers have devised this study. In this study, we will explicitly talk about preprocessing strategies by conducting tests with different combinations of preprocessing techniques. These techniques include data cleansing, stopwords, stemming, and case folding. After that, the recall-oriented understudy for gisting evaluation (ROUGE) assessment will be used to conduct the measurement of the research results. According to the findings of this research, the optimal level of performance may be accomplished by combining the processes of data cleaning and case folding with scores of 0.78, 0.60, and 0.68 for ROUGE-1, ROUGE-2, and ROUGE-L respectively.

Corresponding Author:

Mardhiya Hayaty
Department of Informatics, Faculty of Computer Science, Universitas Amikom Yogyakarta
Ring Road Utara, Depok, Sleman, Yogyakarta, Indonesia
Email: mardhiya_hayati@amikom.ac.id

1. INTRODUCTION

When looking for information, people have to sift through hundreds or even thousands of results on the internet, which is a direct result of the exponential growth in the amount of data. Many informational resources found on the internet call for in-depth research in natural language processing (NLP). Because of this, a system known as automatic text summarization was developed, and it quickly gained popularity, in order to condense information so that it is easier for people to comprehend [1]. Documents that provide summaries of text will be required more frequently to assist in the resolution of these issues.

Automated text summarization (ATS) is a method of extracting the essence of information from text documents and containing the overall meaning of those texts [2]. ATS is now one of the most popular NLP research fields for producing high-quality short paragraphs that cover the major body of a text document. Readability, coherence, syntax, non-redundancy, sentence order, diversity of information, and information coverage are some factors to consider for good summary findings [3]. Automatic summarizing techniques are classified into two types: extractive and abstractive [4], [5]. An extractive summary extracts the most important sections of a document without modifying the wording. Abstractive summaries modify sentences to make new ones conceivable, and the results can be comparable to human summaries. Abstractive summaries are more difficult to write because they involve meaning representation, content arrangement,

surface manifestation, and intuitive understanding [6], [7]. Even though the themes, data types, and algorithms differ, there are several sorts of ATS research. In the blog summarization dataset, extractive research was carried out utilizing the SummCoder algorithm approach [8]. ROUGE-1 (78.0), ROUGE-2 (71.7), ROUGE-SU4 (71.8), and ROUGE-L (72.7) are the results given to the blog summary data set, SummCoder, followed by Com01 and Alg09, with scores of 77.0 and 76.0 ROUGE-1, respectively. The study was conducted in abstract form, with a genetic semantic graph used to summarize Indonesian news [9]. The results showed that a 100-word summary had an average ROUGE-2 (0.32) and a 200-word summary had an average ROUGE-2 (0.39). Study [10] ROUGE-1 (0.11975), ROUGE-2 (0.01199) in scenario 1 with 128 hidden units, and ROUGE-1 (0.06745), ROUGE-2 (0.0055) in scenario 2 with 64 hidden units, are the results of abstractive summarization in Indonesian using BiGRU.

Raw data is prone to noise, missing numbers, and inconsistencies, which degrade the accuracy of the result [11]. Data preprocessing is an important first step in determining data quality. Preprocessing is the process of structuring text documents such that a machine can read them easily. Data preparation is used in every system created for text processing and NLP. According to research findings [12], [13], preprocessing improves system performance. Unfortunately, earlier studies did not discuss the impact of the various preprocessing approaches utilized, and it is also uncertain what kind of preprocessing combination delivers best sentiment analysis performance. As a result, this study will concentrate on using various preprocessing approaches to determine the effect of preprocessing on the version of automatic text summarizers. ROUGE will be used to analyze the results that are produced by the machine summarization. The recall-oriented understudy for gisting evaluation also known as ROUGE [14] is a measure or parameter for evaluation that examines the results of summarizing text texts in an automated fashion.

Combining convenient features and preprocessing stages can improve summary performance and reduce the amount of computation required [15]. Based on these findings, the features and preprocessing tasks used to achieve the best summary performance may differ depending on the text's domain and the success metrics used. As a result, the purpose of this research is to determine the impact of preprocessing techniques on the result of summarization, so that this research can contribute to the influential preprocessing stages and determine which preprocessing techniques are required or not. The following is an outline for this paper. Section 2 describes the research methodology, datasets, and experimental scenarios. Section 3 then explains the experimental results, and section 4 concludes.

2. METHOD

2.1. Research flow

This research aims to examine the performance impact of the pre-trained model by implementing a combination of preprocessing stages and evaluate model performance systematically using the proposed preprocessing technique. It is hoped that this research can contribute to the development of more accurate and efficient natural language models. The system to be created is shown in Figure 1.

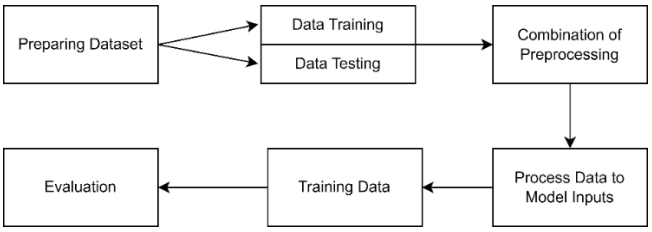

Figure 1. Research flow

In the research flow in Figure 1, the researcher first entered and prepared the dataset. After that, the distribution of training and testing data was carried out. Then carry out 16 experimental combinations of preprocessing stages such as data cleaning, stemming, stopwords, and case folding. After applying the preprocessing stage, the next step is to process the data to the input model and the training process. In the final stage, evaluation and testing will be carried out using test data.

2.2. Transformers – BERT

Transformers architecture has two essential components, namely encoder and decoder. The encoder functions to capture and convert the input sequence into binary form. The Decoder will display the results of

the machine process as output that humans can understand [16]. The two are linked by the attention mechanism, shown in Figure 2.

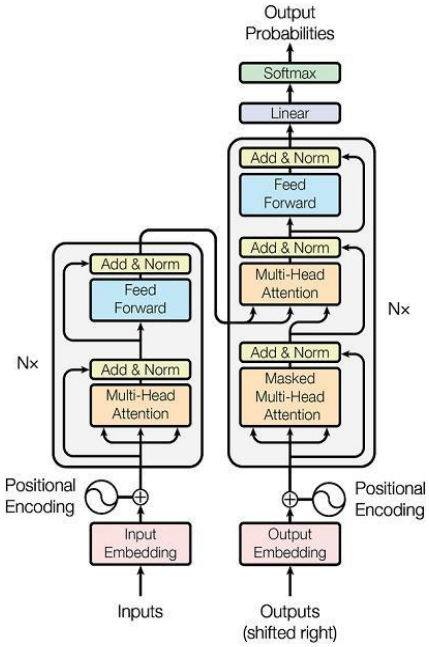

Figure 2. Transformers – model architecture

Bidirectional encoder representations from transformers (BERT) is a transformer-based machine learning technique trained in an English corpus to simplify the pre-training process in NLP [17]. The basic transformer consists of an encoder to read the input text and a decoder to generate predictive assignments. BERT only requires an encoder to produce a language representation model. The BERT architecture has 2 phases of use, namely Pre-training and Fine-Tuning, which can be combined in various tasks. There are slight differences between the pre-trained and final architecture shown in Figure 3. Study [18] introduces IndoBERT, a modified version of BERT Base following the BERT-Base (uncased) configuration. IndoBERT has been trained to use 220 million words using three main sources, namely Indonesia Wikipedia (74M words), Kompas Tempo and Liputan6 articles (55M total) and Indonesia Web Corpus (90M words).

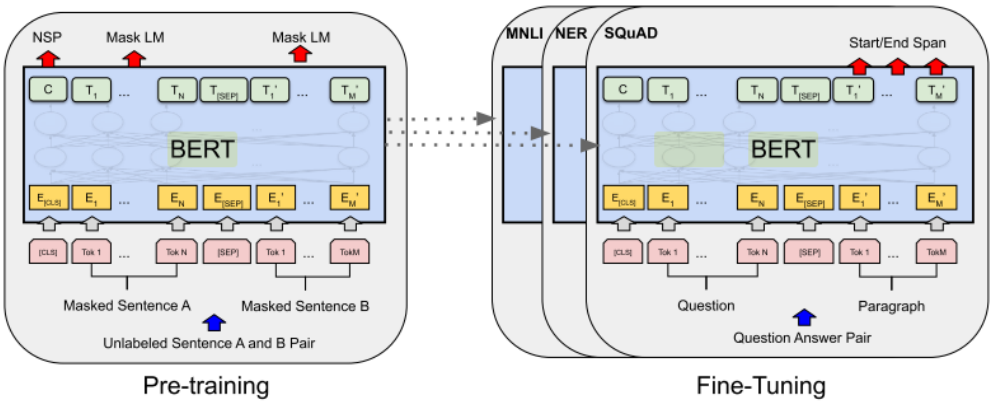

Figure 3. BERT pre-training and Fine-tuning - model architecture

2.3. Preprocessing

This research uses data cleaning, stop words, stemming and case folding at the preprocessing stage. Studies [19]–[23] mentioned that stopword, stemming, and tokenization are the stages most often used to

summarize the text. Tokenization is the process of splitting text into tokens. Words, symbols, numbers, punctuation marks and other essential entities can be considered tokens. Stopword removal aims to extract important words from the token results. Stemming is the process of eliminating reducing the number of index sentences by removing affixes into basic forms. The stopwords and stemming used in this study use the library from Sastrawi. Several combinations will be applied in this study by adding data cleaning and case folding. Data cleaning removes digit numbers from strings, punctuations, URLs, and white spaces. Case folding aims to change all letters in a document to lowercase. The common practice in most automated text summarisation studies is applying all the pre-processing methods without thoroughly analyzing their contribution to summary performance. Therefore, this study involves several experimental scenarios to see whether there is an influence from the preprocessing stage on the summary system built with the four preprocessing techniques listed in Table 1.

Table 1. Experiments of the preprocessing method

Experiment	Data Cleaning	Stop Words	Stemming	Case Folding
1	Enabled	Enabled	Enabled	Enabled
2	Enabled	Enabled	Enabled	Disabled
3	Enabled	Enabled	Disabled	Disabled
4	Enabled	Disabled	Disabled	Disabled
5	Disabled	Enabled	Disabled	Disabled
6	Disabled	Disabled	Enabled	Disabled
7	Disabled	Disabled	Disabled	Enabled
8	Enabled	Disabled	Enabled	Disabled
9	Enabled	Disabled	Disabled	Enabled
10	Disabled	Enabled	Enabled	Disabled
11	Disabled	Enabled	Disabled	Enabled
12	Disabled	Disabled	Enabled	Enabled
13	Enabled	Enabled	Disabled	Enabled
14	Enabled	Disabled	Enabled	Enabled
15	Disabled	Enabled	Enabled	Enabled
16	Disabled	Disabled	Disabled	Disabled

2.4. Model

In the modelling stage using pretrained IndoBERT, a fine-tuning process will be carried out to optimize the model so that it can be used in the summarization process. IndoBERT architecture is shown in Figure 4. It is hoped that by using IndoBERT as the basic model and fine-tuning, the model can produce quality summary text according to user needs.

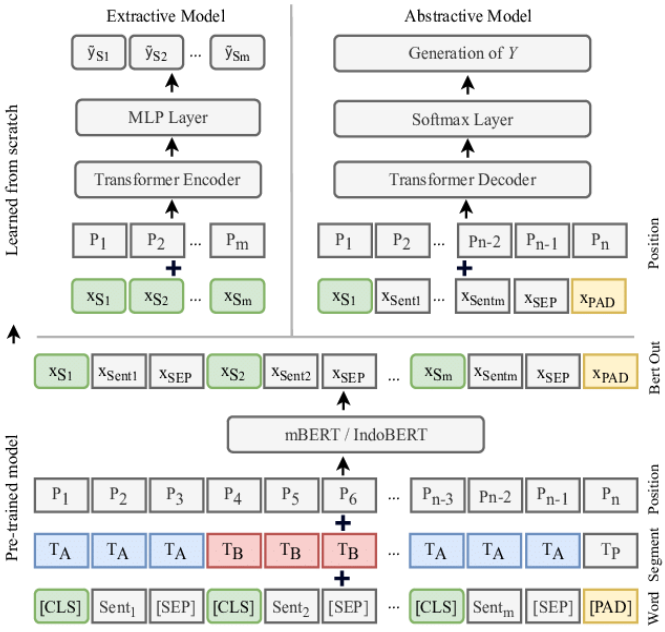

Figure 4. Architecture of the IndoBERT for summarization model

Based on Figure 4. IndoBERT changes the input of a sentence into a token sequence. During the tokenization process, special tokens will be added, namely [CLS], [SEP], and [PAD] tokens. Token [CLS] is a token to symbolize the start of a sentence, and token [SEP] is a token to separate between sentences. [PAD] token to add padding and maximize the initialized token. In implementing text summarization, tokens [CLS] and [SEP] are inserted at the beginning and end of each sentence. A mask carries out bidirectional training with a certain percentage of the input token trained by pretraining. The transformers encoder's and MLP layers' parameters are randomly initialised [24]. The transformer's encoder is configured as follows: layer=2, hidden size=768, feed-forward=2.048, and heads=8. The hyperparameters are trained using the Adam optimizer with a learning rate=$3e^{-5}$, batch size=16, epoch=7, and weight decay=$5e^{-3}$. The hardware specifications used can be described as follows:

- Device Name: Laptop Asus Vivobook A416JA
- RAM: 8 GB
- GPU: Intel UHD Graphics
- CPU: Intel Core i3-1005G1
- Software: Google Colaboratory

2.5. Evaluation

ROUGE [14] is an evaluation metric or parameter that automatically evaluates the results of summarizing text documents. ROUGE evaluates the summary results by comparing the machines and human results (gold summary). The most popular evaluation metrics used for ATS are ROUGE-N, and ROUGE L. ROUGE-N is a recall calculation based on n-grams between gold summary and machine summarized text. The number of n-grams often used is n=1 (ROUGE 1) and n=2 (ROUGE 2). For example, x is the number of n-grams that is the same between the gold standard summary and the machine-summarized text, and y is the number of n-grams in the gold standard summary. Then ROUGE-N can be calculated by the following formula,

$$ROUGE\text{-}N = \frac{x}{y}$$

ROUGE-L evaluates text summaries by comparing the longest common subsequence (LCS) or the longest series of words that are the same between the engine text summary results and the gold standard summary. For example, z is the number of words in the gold standard summary, then ROUGE-L can be calculated using the following formula,

$$ROUGE\text{-}L = \frac{LCS}{z}$$

3. RESULTS AND DISCUSSION

This section reports the results of 16 experiments conducted to assess the accuracy of the summary results before and after applying the preprocessing method. The difference in each scenario is in the preprocessing section. This text summary test uses a dataset from IndoSum [25] of 14,262 news articles divided into 80% train data and 20% validation data. News articles are taken from Indonesian language news portals with titles, categories, and two gold standard summaries made manually. The test data consisting of 3762 articles have been applied to the preprocessing stage according to the experimental scenario used to test the model. From the results of the summary, a text summary performance evaluation will be carried out using the Rouge Score to determine the accuracy of the system being built. Table 2 are the results of the 16 experiments that have been carried out.

Based on Table 2, of the 16 experiments that have been carried out, it turns out that the highest ROUGE score was found in experiment 9 with ROUGE-1 (0.78), ROUGE-2 (0.60), and ROUGE-L (0.68) scores. The best system performance is obtained when combining data cleaning and case folding. The high ROUGE score in experiment 9 is due to the data cleaning process, which cleans dirty data. The application of case folding also has an effect because the data becomes structured and consistent in the use of capital letters. However, case folding without the data cleaning process gets the lowest results as in experiment 7. The lowest ROUGE value in experiment 7 gets ROUGE-1 (0.16), ROUGE-2 (0.05), and ROUGE-L (0.15) scores. The low results in experiment 7 were caused by the absence of a data cleaning process, so the model could not capture the information contained in the original text, or the summary results could have been better. Preprocessing testing using data cleaning produces better performance when compared to testing without using data cleaning, shown in Figure 5.

Table 2. Result of experiments

Experiment	Data Cleaning	Stop Words	Stemming	Case Folding	R1	R2	RL
1	Enabled	Enabled	Enabled	Enabled	0.428	0.218	0.378
2	Enabled	Enabled	Enabled	Disabled	0.446	0.238	0.398
3	Enabled	Enabled	Disabled	Disabled	0.422	0.217	0.380
4	Enabled	Disabled	Disabled	Disabled	0.681	0.561	0.655
5	Disabled	Enabled	Disabled	Disabled	0.344	0.124	0.320
6	Disabled	Disabled	Enabled	Disabled	0.405	0.164	0.387
7	**Disabled**	**Disabled**	**Disabled**	**Enabled**	**0.165**	**0.055**	**0.158**
8	Enabled	Disabled	Enabled	Disabled	0.353	0.264	0.329
9	**Enabled**	**Disabled**	**Disabled**	**Enabled**	**0.708**	**0.603**	**0.685**
10	Disabled	Enabled	Enabled	Disabled	0.299	0.109	0.289
11	Disabled	Enabled	Disabled	Enabled	0.302	0.108	0.291
12	Disabled	Disabled	Enabled	Enabled	0.314	0.119	0.305
13	Enabled	Enabled	Disabled	Enabled	0.550	0.348	0.506
14	Enabled	Disabled	Enabled	Enabled	0.607	0.449	0.579
15	Disabled	Enabled	Enabled	Enabled	0.282	0.098	0.273
16	Disabled	Disabled	Disabled	Disabled	0.370	0.130	0.355

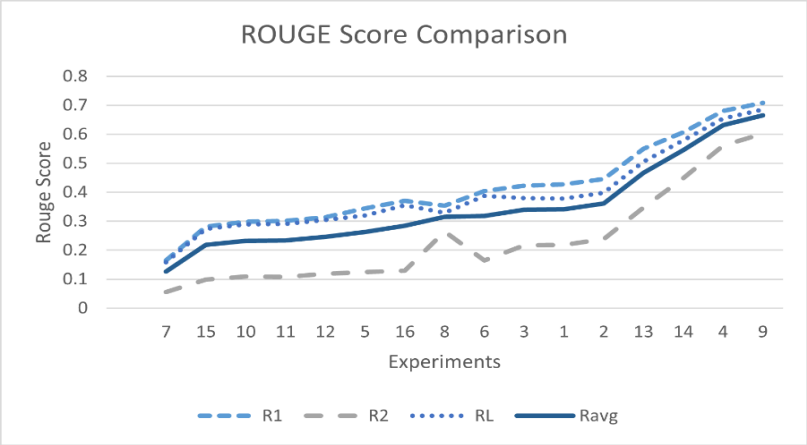

Figure 5. ROUGE score in ascending order

Figure 5 the nine highest experimental trials used data cleaning, except for experiment 6. This experiment only used stemming with an average ROUGE score better than experiment 8, which used a combination of data cleaning and stemming. The stemming used in the experiment cleaned dirty data even though there were still a few dashes, thus influencing the ROUGE score. Experimental combination testing without data cleaning will hurt the accuracy value in the seven lowest experiments. Most tests involving stopwords, stemming, and case folding produces low accuracy. Meanwhile, using stopwords and stemming techniques accompanied by data cleaning has a negative effect even though the accuracy results on the stopwords and stemming tests produce good accuracy. This is because when using stopwords and stemming, there are words that, if omitted, can reduce the information from the sentence so that the features used cannot describe the data. Using a large number of preprocessing techniques does not guarantee better system performance accuracy.

The results of experiment 7, Table 3. is one of the samples in the preprocessing stage that has been applied using only case folding. Before preprocessing, the articles and references summary columns still contained unnecessary punctuation, URLs, digits, and white space. After the preprocessing stage, the data still looks the same as before; only all letters are lowercase. The summary generated by the model is also ugly because the system needs to capture complete information.

From the test results in experiment 9, it can be seen that Table 4 is one of the preprocessing stages samples that have been applied. Before preprocessing, the articles and references summary columns contained unnecessary punctuation, URLs, digits and white space. Data cleaning and case folding are applied to these columns. After the preprocessing stage is carried out, it looks cleaner and easier to read and understand. The summary generated by the model also looks good, with information similar to the reference summary.

Table 3. Sample of summarization result experiment 7

Article:	Preprocessing Article:	References Summary	Preprocessing References Summary	Model Summary
[[[‘Jakarta’, ‘,’, ‘CNN’, ‘Indonesia’, ‘-‘, ‘-‘, ‘Dilansir’, ‘AFP’, ‘,’, ‘seorang’, ‘warga’, ‘Mesir’, ‘yang’, ‘dipercaya’, ‘sebagai’, ‘wanita’, ‘terberat’, ‘di’, ‘dunia’, ‘masuk’, ‘sebuah’, ‘rumah’, ‘sakit’… *([[['Jakarta', ',', 'CNN', 'Indonesia', '-', '-', 'Reported', 'AFP', ',', 'a', 'citizen', 'Egypt' , 'which', 'trusted', 'as', 'woman', 'heaviest', 'in', 'world', 'in', 'a', 'home', 'sick…)*	[[[‘jakarta’, ‘,’, ‘cnn’, ‘indonesia’, ‘-‘, ‘-‘, ‘dilansir’, ‘afp’, ‘,’, ‘seorang’, ‘warga’, ‘mesir’, ‘yang’, ‘dipercaya’, ‘sebagai’, ‘wanita’, ‘terberat’, ‘di’, ‘dunia’, ‘masuk’, ‘sebuah’, ‘rumah’, ‘sakit’… *([[['jakarta', ',', 'cnn', 'indonesia', '-', '-', 'reported', 'afp', ',', 'a', 'citizen', 'egypt' , 'which', 'trusted', 'as', 'woman', 'heaviest', 'in', 'world', 'in', 'a', 'home', 'sick'…)*	[[‘Eman’, ‘Ahmed’, ‘Abd’, ‘El’, ‘Aty’, ‘memiliki’, ‘berat’, ‘badan’, ‘mencapai’, ‘500’, ‘kilogram’, ‘sebelum’, ‘menjalankan’, ‘operasi’, ‘di’, ‘Mumbai’, ‘Maret’, ‘lalu’.. *([['Eman', 'Ahmed', 'Abd', 'El', 'Aty', 'had', 'weight', 'body', 'reached', '500', 'kilogram', 'before', 'run', 'surgery', 'in', 'Mumbai', 'march', 'then'…)*	[‘Eman’, ‘Ahmed’, ‘Abd’, ‘El’, ‘Aty’, ‘memiliki’, ‘berat’, ‘badan’, ‘mencapai’, ‘500’, ‘kilogram’, ‘sebelum’, ‘menjalankan’, ‘operasi’… *('aty', 'had', 'weight', 'body', 'reached', '500', 'kilogram', 'before', 'running', 'surgery'…,)*	[[‘seorang ‘,’warga ‘mesir’ yang’ bernama’ dipercaya’ sebagai’ anita wanita… *(Which' named' trusted' as' anita woman…)*

Table 4. Sample of summarization result experiment 9

Article:	Preprocessing Article	References Summary	Preprocessing References Summary	Model Summary
[[['Merdeka.com', '-', 'Presiden', 'Joko', 'Widodo', '(', 'Jokowi', ')', 'tak', 'hanya', 'membangun', 'rumah', 'untuk', 'pekerja', ',', 'masyarakat', 'berpenghasilan', 'rendah', ',', 'prajurit', 'TNI', '/', 'Polri', 'dan', 'mahasiswa', … *([['Merdeka.com', '-', 'Presiden', 'Joko', 'Widodo', '(', 'Jokowi', ')', 'not', 'only', 'building', ' house', 'for', 'workers', ',', 'community', 'income', 'low', ',', 'soldier', 'TNI', '/', 'Polri', 'and' , 'student',…)*	merdeka com presiden joko widodo jokowi tak hanya membangun rumah untuk pekerja masyarakat berpenghasilan rendah prajurit tni polri dan mahasiswa … *(merdeka com president joko widodo jokowi will not only build houses for low-income community workers, military police and students …)*	[['Presiden', 'Joko', 'Widodo', '(', 'Jokowi', ')', 'akan', 'membangun', 'rusun', 'untuk', 'para', 'santri', 'di', 'pondok-pondok', 'pesantren', '.'],… *([['President', 'Joko', 'Widodo', '(', 'Jokowi', ')', 'will', 'build', 'flat', 'for', 'para', 'students', 'in', 'boarding schools', 'boarding schools', '.'], …)*	presiden joko widodo jokowi akan membangun rusun untuk para santri di pondok pondok pesantren… *(president joko widodo jokowi will build flats for students at Islamic boarding schools…)*	presiden joko widodo tak hanya membangun rumah untuk pekerja masyarakat berpenghasilan rendah prajurit tni polri dan mahasiswa… *(president joko widodo will not only build houses for low-income community workers, military police and students…)*

4. CONCLUSION

A combination of preprocessing data cleaning and case folding produces the best system performance, as determined by the findings of several experimental scenarios for ATS utilizing IndoSum. This can be deduced from the findings as a result of the findings. The ROUGE score can be significantly affected by the thoroughness with which data is cleaned. Case folding, stemming, and stopwords produce results that are not quite as good when applied without prior data cleaning. The use of stemming and stopwords techniques can have a negative impact on the performance of an ATS because these techniques can reduce the amount of information taken from an influential sentence. If you only use a few of the preprocessing techniques, you will ensure the best performance possible for the system. To compare the preprocessing stages of each language, a similar analysis could be run using more than one data set as the basis.

REFERENCES

[1] A. B. Tikarya, K. Mayur, and P. H. Patel, “Pre-processing phase of text summarization based on gujarati language,” *International Journal of Innovative Research in Computer Science & Technology*, vol. ISSN, no. 4, pp. 2347–5552, 2014, [Online]. Available: https://www.researchgate.net/publication/264784466%0APre-Processing.

[2] A. Alomari, N. Idris, A. Q. M. Sabri, and I. Alsmadi, “Deep reinforcement and transfer learning for abstractive text summarization: A review,” *Computer Speech and Language*, vol. 71, 2022, doi: 10.1016/j.csl.2021.101276.

[3] M. Gambhir and V. Gupta, “Recent automatic text summarization techniques: a survey,” *Artificial Intelligence Review*, vol. 47, no. 1, pp. 1–66, 2017, doi: 10.1007/s10462-016-9475-9.

[4] A. K. and N. Salim, “A review on abstractive summarization methods,” *J. Theor. Appl. Inf. Technol.*, vol. 59, no. 1, pp. 64–72, 2016, doi: 10.1109/ICCCNT54827.2022.9984332.

[5] V. Gupta and G. S. Lehal, “A survey of text summarization extractive techniques,” *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no. 3, pp. 258–268, 2010, doi: 10.4304/jetwi.2.3.258-268.

[6] J. ge Yao, X. Wan, and J. Xiao, “Recent advances in document summarization,” *Knowledge and Information Systems*, vol. 53, no. 2, pp. 297–336, 2017, doi: 10.1007/s10115-017-1042-4.

[7] Y. D. Prabowo, A. I. Kristijantoro, H. L. H. S. Warnars, and W. Budiharto, “Systematic literature review on abstractive text summarization using kitchenham method,” *ICIC Express Letters, Part B: Applications*, vol. 12, no. 11, pp. 1075–1080, 2021, doi:

10.24507/icicelb.12.11.1075.
[8] A. Joshi, E. Fidalgo, E. Alegre, and L. Fernández-Robles, "SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders," *Expert Systems with Applications*, vol. 129, pp. 200–215, 2019, doi: 10.1016/j.eswa.2019.03.045.
[9] R. S. Devianti and M. L. Khodra, "Abstractive summarization using genetic semantic graph for indonesian news articles," *Proceedings - 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, ICAICTA 2019*, 2019, doi: 10.1109/ICAICTA.2019.8904361.
[10] R. Adelia, S. Suyanto, and U. N. 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.
[11] R. Reztaputra and M. L. Khodra, "Sentence structure-based summarization for Indonesian news articles," *Proceedings - 2017 International Conference on Advanced Informatics: Concepts, Theory and Applications, ICAICTA 2017*, 2017, doi: 10.1109/ICAICTA.2017.8090983.
[12] V. V. Nhlabano and P. E. N. Lutu, "Impact of text pre-processing on the performance of sentiment analysis models for social media data," *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems, icABCD 2018*, 2018, doi: 10.1109/ICABCD.2018.8465135.
[13] A. Krouska, C. Troussas, and M. Virvou, "The effect of preprocessing techniques on Twitter sentiment analysis," *IISA 2016 - 7th International Conference on Information, Intelligence, Systems and Applications*, 2016, doi: 10.1109/IISA.2016.7785373.
[14] G. Tsuchiya, "ROUGE: A Package for Automatic Evaluation of Summaries Chin-Yew," *Japanese Circulation Journal*, vol. 34, no. 12, pp. 1213–1220, 1871, doi: 10.1253/jcj.34.1213.
[15] S. Bal and E. Sora Gunal, "The impact of features and preprocessing on automatic text summarization," *Romanian Journal of Information Science and Technology*, vol. 25, no. 2, pp. 117–132, 2022.
[16] A. Vaswani *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-December, pp. 5999–6009, 2017.
[17] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, 2019.
[18] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, "IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP," *COLING 2020 - 28th International Conference on Computational Linguistics, Proceedings of the Conference*, pp. 757–770, 2020, doi: 10.18653/v1/2020.coling-main.66.
[19] A. P. Widyassari *et al.*, "Review of automatic text summarization techniques and methods," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 4, pp. 1029–1046, 2022, doi: 10.1016/j.jksuci.2020.05.006.
[20] F. Horasan and B. Bilen, "Extractive text summarization system for news texts," *International Journal of Applied Mathematics Electronics and Computers*, pp. 179–184, 2020, doi: 10.18100/ijamec.800905.
[21] M. Bidoki, M. R. Moosavi, and M. Fakhrahmad, "A semantic approach to extractive multi-document summarization: Applying sentence expansion for tuning of conceptual densities," *Information Processing and Management*, vol. 57, no. 6, 2020, doi: 10.1016/j.ipm.2020.102341.
[22] R. He, J. Tang, P. Gong, Q. Hu, and B. Wang, "Multi-document summarization via group sparse learning," *Information Sciences*, vol. 349–350, pp. 12–24, 2016, doi: 10.1016/j.ins.2016.02.032.
[23] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing and Management*, vol. 50, no. 1, pp. 104–112, 2014, doi: 10.1016/j.ipm.2013.08.006.
[24] F. Koto, J. H. Lau, and T. Baldwin, "Liputan6: A large-scale Indonesian dataset for text summarization," 2020, [Online]. Available: http://arxiv.org/abs/2011.00679.
[25] K. Kurniawan and S. Louvan, "IndoSum: A new benchmark dataset for Indonesian text summarization," *Proceedings of the 2018 International Conference on Asian Language Processing, IALP 2018*, pp. 215–220, 2019, doi: 10.1109/IALP.2018.8629109.

BIOGRAPHIES OF AUTHORS

Mardhiya Hayaty is currently working as Assistant Professor, at Informatics Department in Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia. Her research interest is natural language processing, sentiment analysis, and automatic text summarization. She can be contacted at email: mardhiya_hayati@amikom.ac.id.

Muhammad Farhan Juna currently studying at Amikom Yogyakarta University. His research interest is automatic text summarization. He can be contacted at email: muhammad.juna@students.amikom.ac.id.