Exhibition and Seminar on Science and Creative Technology, University of Al-Azhar Indonesia (EXSACT-A 2024) Proceeding

DOI: http://dx.doi.org/10.36722/exc.v2i1.3378

Development of a Keyword Extractor Using the Bidirectional Encoder Representations from Transformers (BERT) Model

Muhammad Mubarok Azzam[1], Ade Jamal[1]

[1]Department of Informatics, Faculty of Science and Technology, Universitas Al Azhar Indonesia, Sisingamangaraja, South Jakarta, 12110

Corresponding author/E-mail: mmazzam4@gmail.com

Abstract — **To extract key information from documents, keyword extraction is often used as an automated process to identify the most relevant words and phrases. Models like Rapid Automatic Keyword Extraction (RAKE) and Yet Another Keyword Extractor (YAKE) operate based on the statistical properties of text without considering semantic similarity. Bidirectional Encoder Representations from Transformers (BERT), a bidirectional transformer model, addresses this limitation by converting phrases and documents into vectors that capture semantic meaning. This research tests a keyword extraction system on the abstract texts of Indonesian theses using the BERT model "cahya/bert-base-indonesian-1.5G" from HuggingFace. Additionally, the study employs three similarity matrix formulas (Cosine Similarity, Euclidean Distance, Manhattan Distance) to measure the similarity between the text and candidate keywords. The results show that the YAKE model performed best overall, followed by RAKE. The BERT model showed lower performance, but Euclidean Distance for BERT outperformed Cosine Similarity and Manhattan Distance.**

Keywords - *Keyword extraction, abstract text of Indonesian theses, RAKE, YAKE, BERT.*

INTRODUCTION

Keywords are commonly used in natural language processing and information indexing to help in document understanding. The words or phrases generated will describe the content of the information contained in the text [1]. The extraction process is a method for obtaining important points from data [1]. Keyword extraction is a crucial step in analyzing and summarizing information from text.

Models like Rapid Automatic Keyword Extraction (RAKE) and Yet Another Keyword Extractor (YAKE) are examples of models used to extract keywords and key phrases. However, these models generally work based on the statistical properties of the text and do not rely on semantic similarity [2]. To overcome this limitation, the author attempts to create keyword extraction using the Bidirectional Encoder Representations from Transformers (BERT) model.

BERT is a bidirectional transformer model that allows phrases and documents to be converted into vectors that capture their semantic meaning [2]. The BERT model itself applies a limited transformer architecture (encoder-only) used for Natural Language Understanding (NLU) tasks with input in the form of text data and output in the form of vectors representing the entire input along with its context. The BERT model's process has two stages: pre-training and fine-tuning. The BERT model used in this study only involves the pre-training stage. The author uses a pre-trained BERT model to generate text representations in the form of matrices, which are the output of the pre-training stage. This study uses the BERT model obtained from the HuggingFace website with the model name "cahya/bert-base-indonesian-1.5G". The reason for using this model is that it has been pre-trained with 522MB of Indonesian Wikipedia and 1GB of Indonesian news articles and is an "uncased" model, meaning it does not differentiate between uppercase and lowercase letters.

In developing keyword extraction with the BERT model, this study also employs similarity matrix calculations using three formulas: Cosine Similarity, Euclidean Distance, and Manhattan Distance. Cosine Similarity is a method for measuring how similar two vectors are in a multi-dimensional space [3]. Euclidean Distance is a method for measuring the "straight-line" distance between two points in Euclidean space [4]. Manhattan Distance is a method for measuring the distance between two points in a multi-dimensional space by summing the absolute differences of their components [5]. The purpose of using similarity matrix calculations is to assess the similarity between the text and the candidate keywords generated by the BERT model in the form of matrices.

Based on this background, the author is interested in understanding whether keyword extraction using similarity matrix calculations from the BERT model produces better or worse keyword extraction results compared to the RAKE or YAKE models. Additionally, the study aims to understand how the algorithms of these three models work and the architecture of the transformer, which forms the foundation for building BERT.

The evaluation results will be presented in graphical form to provide a better visualization of the performance of each model in keyword extraction. The evaluation is conducted using Recall values. Recall is used to determine the percentage of correct keywords produced by the model compared to the original keywords. Then, the average Recall will be calculated and presented in graphical form to observe the average Recall results for each model.

METHOD

This study uses abstract text data from Indonesian theses. The data was obtained from the website "repoperpus.uai.ac.id", consisting of abstract texts and keywords from student theses at Universitas Al-Azhar Indonesia, spanning from 2023 to 2024. The abstract texts were captured using the Snipping Tool to convert them into images, which were then uploaded to Yandex to be converted into text and saved in a spreadsheet format. A total of 1,357 abstract texts were used in this study.

The data was processed using Google Colaboratory, utilizing the pandas library. The data previously stored in a spreadsheet format was downloaded as a .csv file and stored in Google Drive.

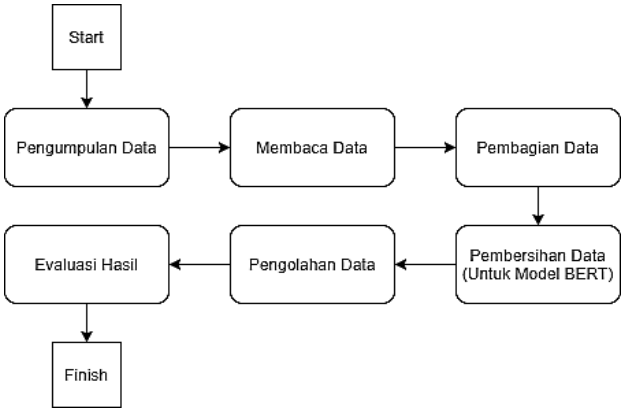

Figure 1. Research methodology

The data was then grouped based on the length of the keywords in the original text, and a dataframe was created for each group. This grouping process resulted in 12 groups, but only 5 groups were used in the study, as the number of data points in groups 6 to 12 was very small.

The data was then cleaned by converting uppercase letters to lowercase, removing numbers, special characters, and excess spaces using the regular expression (re) library and the Natural Language Toolkit (NLTK). Since the RAKE and YAKE models have their own data cleaning processes, the cleaning was not applied to these models.

The data processing began by installing several libraries such as RAKE, YAKE, and sentence_transformers, as well as the BERT model from Hugging Face, specifically the "cahya/bert-base-indonesian-1.5G" model. The grouped data was then processed using the RAKE, YAKE, and BERT models.

For the RAKE and YAKE models, the author used the *ParameterGrid* library from Scikit-learn to determine the best parameter values. The selection of the best parameters was done by calculating the average Recall of the extracted keywords based on the original keywords.

In the RAKE model, the generated_stopwords_percentile parameter was used to determine the most frequently appearing words in the text to be ignored based on a specified percentile value, where words above this percentile (0-100) would be considered candidates for stopwords. Additionally, the generated_stopwords_min_freq parameter calculated words that appeared in the text and were considered stopwords based on the minimum frequency in the distribution.

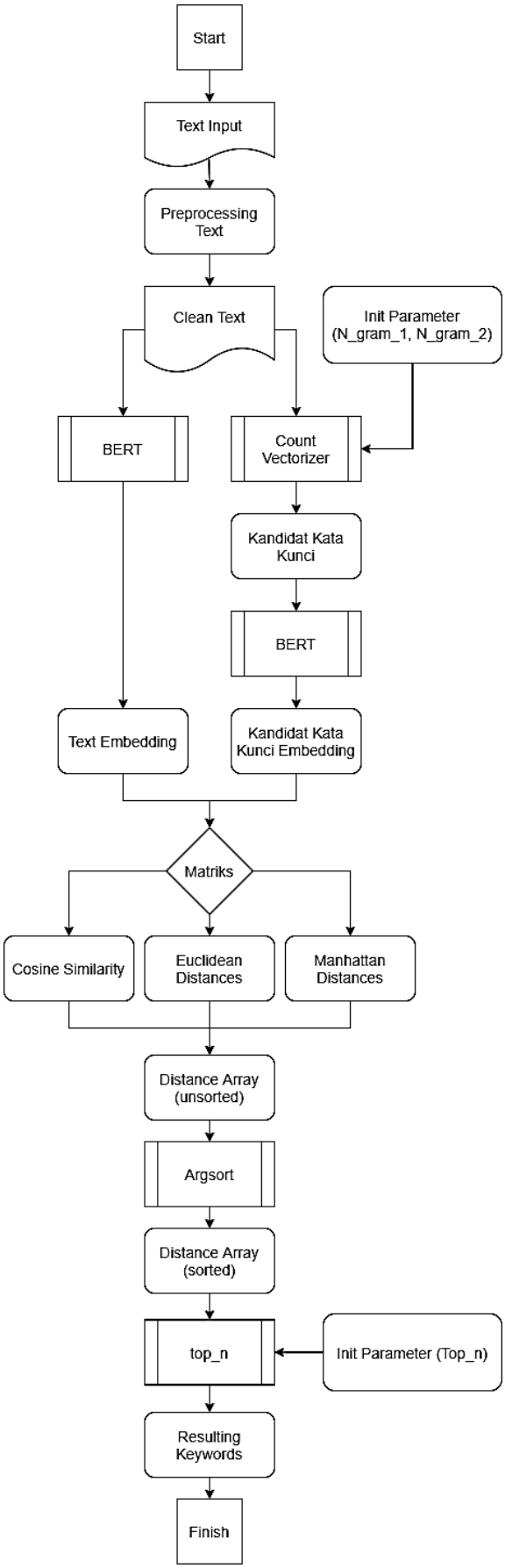

Figure 2. Flowchart with three BERT models

For the YAKE model, the dedupLim parameter controlled the level of similarity considered as duplicates, while the windowsSize parameter set how many words around the keyword would be considered as context. Other parameters not using *ParameterGrid* were adjusted according to the needs of the study.

In this study, the author developed three variants of the BERT model using similarity matrix calculations: Cosine Similarity, Euclidean Distance, and Manhattan Distance. The goal of using these matrix calculation methods was to measure the level of similarity between the candidate text and the abstract text.

The text input consists of Indonesian abstract texts. The preprocessing or text cleaning process aims to convert all text to lowercase, remove all numbers, special characters, unnecessary spaces, and stopwords that are not needed for the analysis. The clean text represents the abstract text data after the cleaning process. In this model, the parameters n_gram_1 and n_gram_2 were used to determine the minimum and maximum keyword lengths to be generated and applied in the *CountVectorizer* function.

CountVectorizer is a module from scikit-learn used to generate keyword candidates by converting the text into a feature matrix based on word or n-gram frequency. The steps of the *CountVectorizer* module include tokenization, vocabulary building, and feature matrix creation. Tokenization divides the text into smaller units called tokens, while vocabulary building creates a list of words or n-grams in the text based on the n_gram_range. After that, the feature matrix is created by counting the occurrences of each token in the text. The get_feature_names_out method is used to view the list of features (vocabulary) extracted from the text. The vocabulary produced by *CountVectorizer* is a collection of all unique words or n-grams found in the given text. The get_feature_names_out method is used to view the list of features (vocabulary) that have been extracted from the text.

In the BERT model with Cosine Similarity, cosine similarity is used to measure the similarity between two vectors: the text and the candidate keywords. This calculation is based on the cosine angle between two vectors in vector space, with values ranging from -1 to 1, where 1 indicates perfect similarity.

The BERT model with Euclidean Distance uses Euclidean distance to measure the distance between two vectors: the text and the candidate keywords. The Euclidean distance is calculated as the straight-line distance between two points in vector space, where the smaller the distance, the more similar the two vectors are.

In the BERT model with Manhattan Distance, Manhattan distance is used to measure the distance between two vectors: the text and the candidate keywords. This calculation is based on the total absolute distance between the coordinates of the text vector and the candidate keywords, with smaller values indicating higher similarity.

The index array of the resulting keyword embedding matrix was sorted from the smallest to largest value using the argsort function. This sorting was done to determine the rank of the keywords based on the calculated embedding values.

The top_n parameter was used to obtain the top_n candidates with certain similarity values. In the Cosine Similarity calculation, keyword selection was based on the highest similarity value, meaning the smallest angle between two vectors: the document embedding and the candidate embedding. In the Euclidean Distance and Manhattan Distance calculations, keyword selection was based on the smallest similarity value, meaning the closest distance between two points: the document embedding and the candidate embedding. The smaller the distance, the closer and more similar the two points are.

The evaluation process was conducted after the data processing with each model was completed. The extracted keywords from each model were combined according to their model to simplify the analysis. The evaluation aimed to calculate the average Recall.

The evaluation process began by comparing the original keywords with the extracted keywords from each model. In the data evaluation procedure, the extracted keywords from each model (RAKE, YAKE, BERT with Cosine Similarity, Euclidean Distance, and Manhattan Distance) were combined according to the model used to facilitate the analysis process. Recall was calculated as the ratio between the number of correct or matching keywords and the total original keywords [1].

$$Recall = \frac{(\text{Model Result Keywords} \cap \text{Original Keywords})}{\text{Total Original Keywords}} \quad (1)$$

The Recall value was calculated by comparing the number of keywords found in both sets (the model results and the original keywords) with the total original keywords. Afterward, the average Recall was calculated and presented in a graph to visualize the average Recall results of each model.

RESULTS AND DISCUSSIONS

In this study, the author conducted four tests. In the first test, the data processing was conducted under the following conditions: abstract texts were selected with a length ranging from 100 to 500 words, resulting in a total of 1,322 abstract texts. The RAKE and YAKE models used *ParameterGrid* to determine the best parameters. Text cleaning for the BERT model also included the removal of stopwords. This first test was divided into three groups based on the number of keywords used: the first group used 5 keywords, the second group used 10 keywords, and the third group used 20 keywords.

Table 1 and Figure 3 are showing the average Recall evaluation results for each model in the first test for the first group.

Table 1. Recall from the first test in the first group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	26.44	33.78	12.24	13.99	13.30
Highest Recall (%)	100	100	100	100	100

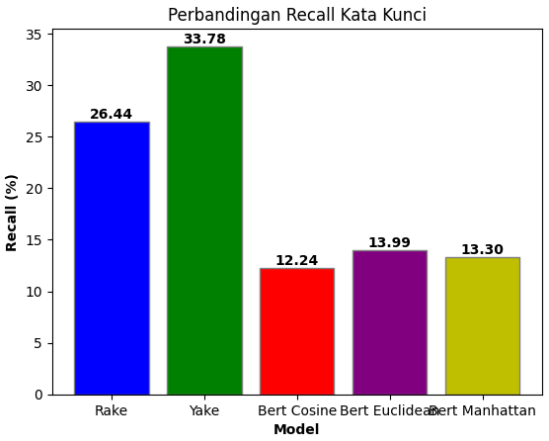

Figure 3. Results from first test in the first group

Table 2 and Figure 4 are showing the average Recall evaluation results for each model in the first test for the second group.

Table 2. Recall from first test in the second group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	39.02	47.89	20.83	22.73	22.73
Highest Recall (%)	100	100	100	100	100

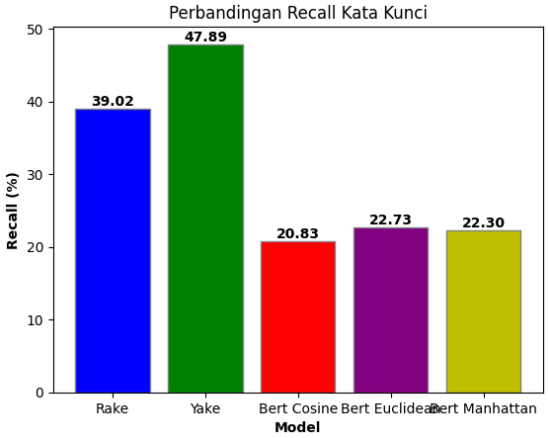

Figure 4. Results from first test in the second group

Table 3 and Figure 5 are showing the average Recall evaluation results for each model in the first test for the third group.

Table 3. Recall from the first test in the third group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	46.37	60.59	34.92	37.92	36.90
Highest Recall (%)	100	100	100	100	100

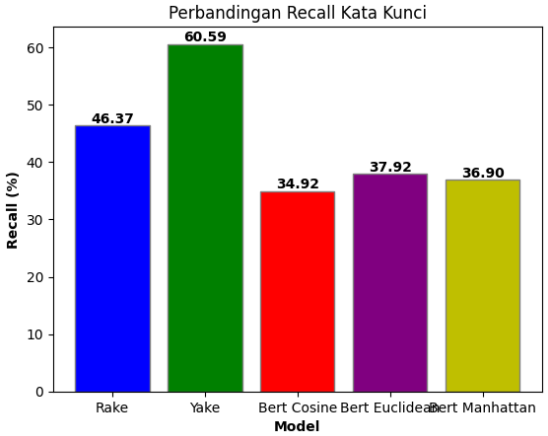

Figure 5. Results from first test in the third group

In the second test, the data processing followed these rules: abstract texts were selected with a length ranging from 100 to 500 words, resulting in a total of 1,322 abstract texts. The parameters for the RAKE and YAKE models used default values or values pre-determined by the models. Text cleaning for the BERT model included the removal of stopwords. This second test was divided into three groups based on the number of keywords used: the first group used 5 keywords, the second group used 10 keywords, and the third group used 20 keywords.

Table 4 and Figure 6 are showing the average Recall evaluation results for each model in the second test for the first group.

Table 4. Recall from the second test in the first group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	26.44	32.57	12.24	13.99	13.30
Highest Recall (%)	100	100	100	100	100

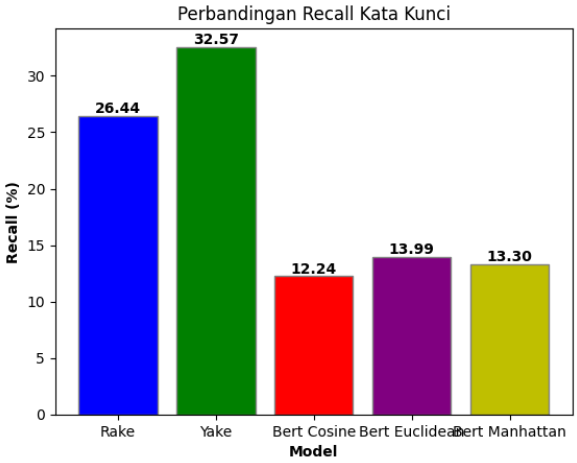

Figure 6. Results from second test in the first group

Table 5 and Figure 7 are showing the average Recall evaluation results for each model in the second test for the second group.

Table 5. Recall from the second test in the second group.

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	39.02	47.55	20.83	22.73	22.30
Highest Recall (%)	100	100	100	100	100

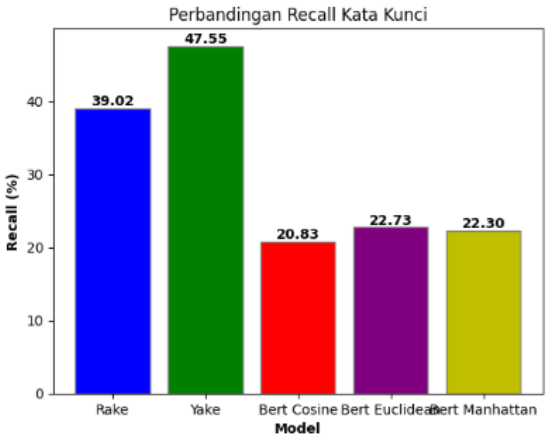

Figure 7. Results from second test in the second group

Table 6 and Figure 8 are showing the average Recall evaluation results for each model in the second test for the third group.

Table 6. Recall from the second test in the third group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	46.37	60.47	34.92	37.92	36.90
Highest Recall (%)	100	100	100	100	100

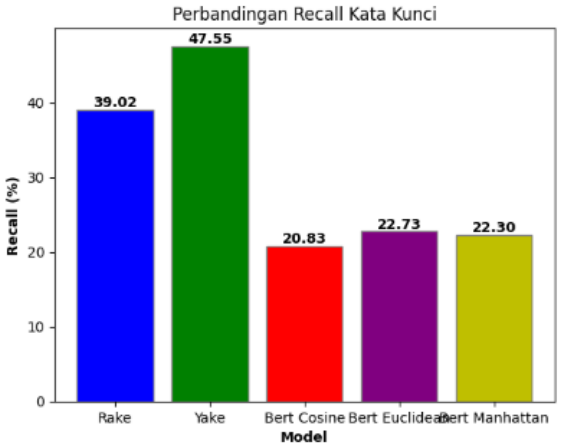

Figure 8. Result from the second test in the third group

In the third test, the data processing followed these rules: abstract texts were not selected or filtered. The RAKE and YAKE models used *ParameterGrid* to determine the best parameters. Text cleaning for the BERT model did not include the removal of stopwords. This third test was divided into three groups based on the number of keywords used: the first group used 5 keywords, the second group used 10 keywords, and the third group used 20 keywords.

Table 7 and Figure 9 are showing the average Recall evaluation results for each model in the third test for the first group.

Table 7. Recall from the third test in the first group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	26.77	33.85	10.08	11.97	10.56
Highest Recall (%)	100	100	100	100	100

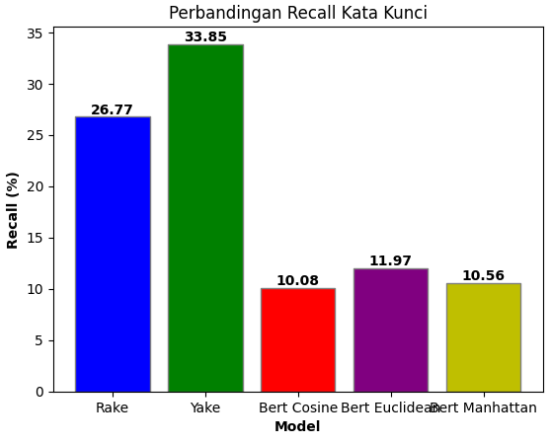

Figure 9. Results from the third test in the first group

Table 8 and Figure 10 are showing the average Recall evaluation results for each model in the third test for the second group.

Table 8. Recall from the third test in the second group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	39.18	48.15	17.03	20.43	19.20
Highest Recall (%)	100	100	100	100	100

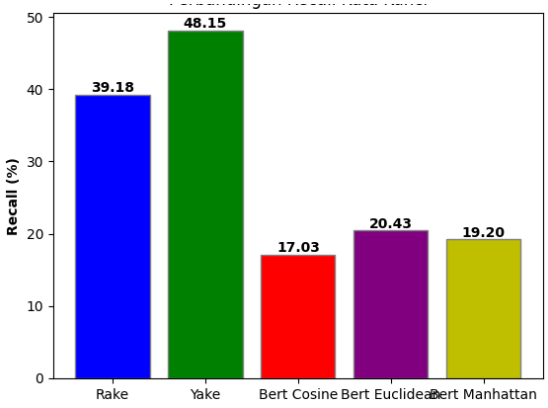

Figure 10. Results from the third test in the second group

Table 9 and Figure 11 are showing the average Recall evaluation results for each model in the third test for the third group.

Table 9. Recall results from the third test in the third group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	46.38	60.91	29.03	33.99	32.59
Highest Recall (%)	100	100	100	100	100

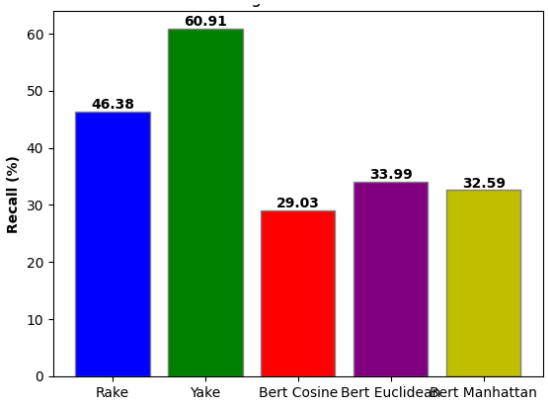

Figure 11. Recall from the third test in the third group

In the fourth test, the data processing followed these rules: abstract texts were not selected or filtered. The parameters for the RAKE and YAKE models used default values or values pre-determined by the models. Text cleaning for the BERT model did not include the removal of stopwords. This fourth test was divided into three groups based on the number of keywords used: the first group used 5 keywords, the second group used 10 keywords, and the third group used 20 keywords.

Table 10 and Figure 12 are showing the average Recall evaluation results for each model in the fourth test for the first group.

Table 11 and Figure 13 are showing the average Recall evaluation results for each model in the fourth test for the second group.

Table 10. Recall from the fourth test in the first group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	26.77	32.70	10.08	11.97	10.56
Highest Recall (%)	100	100	100	100	100

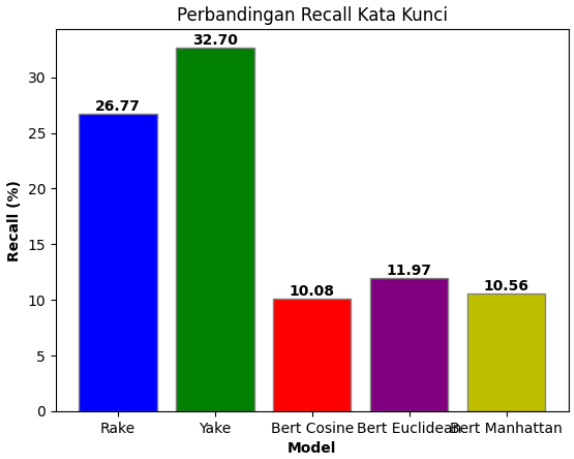

Figure 12. Recall from the fourth test in the first group

Table 11. Recall from the fourth test in the second group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	39.18	47.77	17.03	20.43	19.20
Highest Recall (%)	100	100	100	100	100

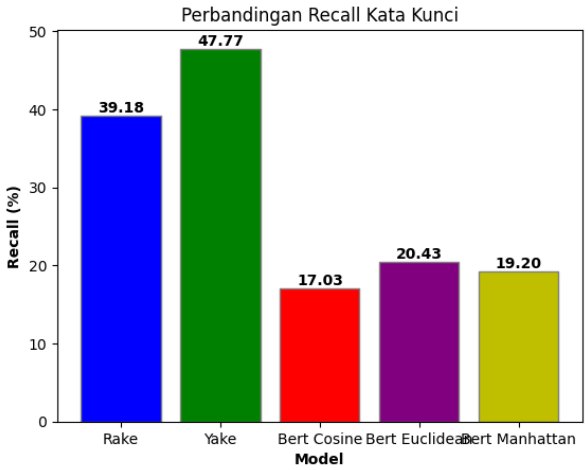

Figure 13. Results from the fourth test in the second group

Table 12 and Figure 14 are showing the average Recall evaluation results for each model in the fourth test for the third group.

Table 12. Recall results from the fourth test in the third group

	RAKE Model	YAKE Model	BERT (Cosine Sim. Mtrx)	BERT (Euclid. Dist. Matrix)	BERT Model (Manhattan Dist. Matrix)
Average Recall (%)	46.38	60.73	29.03	33.99	32.59
Highest Recall (%)	100	100	100	100	100

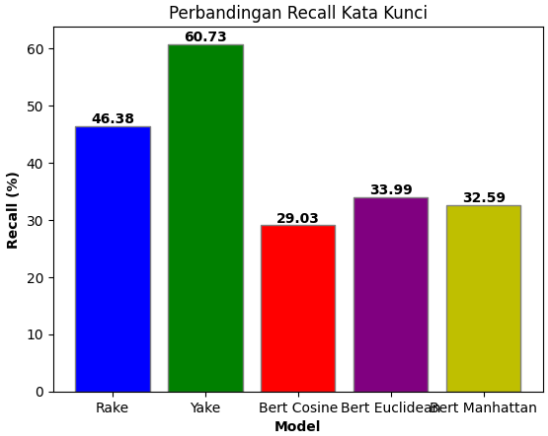

Figure 14. Results from the fourth test in the third group

The use of the *ParameterGrid* module to find the best parameters in the YAKE model showed a slight improvement in the average Recall value compared to using default values or values pre-determined by the model. However, this improvement was not seen in the RAKE model, where no significant changes in the average Recall value were observed. This indicates that parameter adjustment using *ParameterGrid* in the YAKE model can improve performance, although the improvement is not significant. Parameter optimization helped the YAKE model better adapt to the characteristics of the tested data, resulting in more accurate performance. For example, in the first test of the third group, the average Recall value for the YAKE model was 60.59, slightly higher than in the second test of the third group, where the Recall value for the YAKE model was 60.47 without using *ParameterGrid*.

The selection of abstract texts with a length between 100 and 500 words showed a slight increase in the average Recall for both the RAKE and YAKE models. However, for the BERT model with various similarity metrics, the average Recall slightly decreased. This was due to differences in the amount of data used, where before selection, the data totaled 1,364, but after selection, the data was reduced to 1,322. For example, in the second test of the first group, the average Recall value for the RAKE model was 26.44, slightly higher than in the fourth test of the first group, where the RAKE model achieved a Recall value of 26.77 without data selection.

From the test results, it can be concluded that increasing the number of keywords consistently results in higher Recall for all models. This indicates that using more keywords can help capture more relevant information from the document.

The addition of more keywords allows the model to have a broader coverage in information extraction, reducing the chances of missing important information. For example, in the second test of the first group with 5 keywords, the Recall value for the YAKE model was 32.57. In the second test of the second group with 10 keywords, the Recall value for the YAKE model increased to 47.55. In the second test of the third group with 20 keywords, the Recall value for the YAKE model reached 60.47. The increase in the number of keywords consistently raised the average Recall value.

Adding a stopwords cleaning process for the BERT model with various similarity metrics resulted in a higher average Recall compared to without stopwords cleaning. Cleaning stopwords helped reduce noise in the text data, allowing the model to focus on more meaningful and relevant words. This shows that removing stopwords is an important step in text preprocessing to improve the performance of the BERT model in keyword extraction. For example, in the second test of the first group, the Recall value for the BERT model with Cosine Similarity was 12.24, slightly higher than the Recall value in the fourth test of the first group for the BERT model with Cosine Similarity, which was 10.08 without text cleaning.

CONCLUSION

The YAKE model proved to be the best overall model, with a high average Recall in every test. This shows that YAKE is the most reliable model for keyword identification. The RAKE model also showed good performance with relatively high average Recall in each test, indicating that RAKE can perform well under certain conditions.

Meanwhile, the BERT model with various metrics (Cosine Similarity, Euclidean Distance, Manhattan Distance) did not show satisfactory results in each test. The proposed development process involved two main stages, namely the candidate keyword extraction process using the *CountVectorizer* module and the embedding process with BERT, but the results showed overall lower performance. It is possible that the *CountVectorizer* module used for candidate keyword extraction caused the unexpected performance. Nonetheless, the BERT model with Euclidean Distance showed a slight advantage over Cosine Similarity and Manhattan Distance in each test.

REFERENCES

[1] M. A. Shiddiq. Ekstraksi kata kunci pada artikel menggunakan metode TextRank S, UIN Malang. 2019.

[2] M. Grootendorst. Keyword extraction with BERT. Maartengrootendorst. 2020. https://www.maartengrootendorst.com/blog/keybert. (Diakses pada 10 Juli 2024).

[3] C. D. Manning, P. Raghavan & H. Schütze. An introduction to information retrieval. Cambridge University. 2009.

[4] P. Barret. Euclidean distance: Raw, normalized, and double-scaled coefficients. PBarret.net. 2005.

[5] H. S. Ranjitkar & S. Karki. Comparison of A*, Euclidean and Manhattan distance using Influence Map in Ms. Pac-Man. 2016.

[6] Y. Matsuo & M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. Int. J. Artif, Vol. 13, no. 1, pp. 157-169. 2004.

[7] R. Mihalcea & P. Tarau. TextRank: Bringing Order into Text. Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, pp. 404-411. 2004.

[8] A. Jain, G. Kulkarni, & V. Shah. Natural language processing. International Journal of Computer Sciences and Engineering, Vol. 6, No. 1, pp. 161-167. 2018.

[9] M. Emms & S. Luz. Machine Learning for Natural Language Processing. ESSLLI 2007 Course Reader. 2007.

[10] M.V. Koroteev. BERT: A Review of Applications in Natural Language Processing and Understanding. 2021. arXiv:2103.11943

[11] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub & W. Pedrycz. A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. Expert Systems with Applications, Vol. 241, No. 122666. 2023.

[12] J. Devlin, M. Chang, K. Lee, & K. Toutanova. BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding. North American Chapter of the Association for Computational Linguistics. pp. 4171–4186. 2019.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser & I. Polosukhin. Attention is All you Need. Neural Information Processing Systems. 2017.

[14] M. H. Sazli. A brief review of feed-forward neural networks. Commun.Fac.Sci.Univ.Ank.Series A2-A3: Phys.Sci. and Eng, Vol. 50, No. 01. 2006.

[15] D. Park & C. W. Ahn. Self-Supervised Contextual Data Augmentation for Natural Language Processing. Symmetry, Vol. 11, No. 11, pp. 1393. 2019.

[16] A. Mittal & A. Modi. ReCAM@IITK at SemEval-2021 Task 4: BERT and ALBERT based Ensemble for Abstract Word Prediction. International Workshop on Semantic Evaluation (SemEval2021), Online. 2021.

[17] T. Setyorini. e-Modul matematika kelas XI: Matriks. Repositori Institusi Kementrian Pendidikan, Kebudayaan, Riset, dan Teknologi. 2019.

[18] H. M. Abdallah, A.Taha & M. M. Selim. Cloud-Based Fuzzy Keyword Search Scheme Over Encrypted Documents. Int. J. Sociotechnology Knowl, Vol. 13, No. 4, pp. 82-100. 2021.

[19] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes & A. Jatowt. A Text Feature Based Automatic Keyword Extraction Method for Single Documents. European Conference on Information Retrieval (ECIR), zrenoble, France. 2018.

[20] M. Chaudhary. TF-IDF vectorizer scikit-learn. Medium. 2020. https://medium.com/@cmukesh8688/tf-idf-vectorizerscikit-learn-dbc0244a911a. (Diakses pada 11 Juli 2024).

[21] J. Alammar. The illustrated transformer. Jalammar.github.io. 2018. https://jalammar.github.io/illustrated-transformer. (Diakses pada 11 Juli 2024).