International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 5, October 2023, pp. 5472~5482
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i5.pp5472-5482

Enhanced TextRank using weighted word embedding for text summarization

Evi Yulianti, Nicholas Pangestu, Meganingrum Arista Jiwanggi
Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia

Article Info

Article history:

Received Feb 4, 2023
Revised Mar 15, 2023
Accepted Apr 7, 2023

Keywords:

Bidirectional encoder representations from transformers
FastText
Term frequency-inverse document frequency
Text summarization
TextRank
Weighted word embedding
Word2Vec

ABSTRACT

The length of a news article may influence people's interest to read the article. In this case, text summarization can help to create a shorter representative version of an article to reduce people's read time. This paper proposes to use weighted word embedding based on Word2Vec, FastText, and bidirectional encoder representations from transformers (BERT) models to enhance the TextRank summarization algorithm. The use of weighted word embedding is aimed to create better sentence representation, in order to produce more accurate summaries. The results show that using (unweighted) word embedding significantly improves the performance of the TextRank algorithm, with the best performance gained by the summarization system using BERT word embedding. When each word embedding is weighed using term frequency-inverse document frequency (TF-IDF), the performance for all systems using unweighted word embedding further significantly improve, with the biggest improvement achieved by the systems using Word2Vec (with 6.80% to 12.92% increase) and FastText (with 7.04% to 12.78% increase). Overall, our systems using weighted word embedding can outperform the TextRank method by up to 17.33% in ROUGE-1 and 30.01% in ROUGE-2. This demonstrates the effectiveness of weighted word embedding in the TextRank algorithm for text summarization.

Corresponding Author:

Nicholas Pangestu
Faculty of Computer Science, Universitas Indonesia
Depok, West Java, Indonesia
Email: pangestu.nicholas@gmail.com

1. INTRODUCTION

According to the Datareportal statistics in 2022 [1], Indonesia is a country where 204 million (73%) of the population use internet. Every day, Indonesians spend an average of 20 percent of their internet time reading the news. Their interest in reading the news is also supported by the increasing number of Indonesian online news portals, which is estimated to reach 43,300 media based on the Press Council statistics in 2017 presented in [2]. Even though the public's interest in reading the news is quite large, reading long stories can take a long time. It then may discourage them to read through the news, especially because people in the digital era tend to expect to find information from various sources quickly [3]. As a result of this tendency, we often see the term "too long; didn't read" or "tl:dr" which is an outline or summary of writings that are spread on the internet. This term is aimed to let other people quickly understand the main point of the text. The presence of the term "tl;dr" itself also shows that the length of a text affects one's reading interest.

It is beneficial if a lengthy text can be condensed into a more compact form, i.e., summary, so that it could save time and effort of readers to find important information from the text [4]. According to Radev *et al.* [5], a summary is a text produced from one or more texts, which conveys important information

from them and the length is normally less than half the length of the original text. A specific research field that explores the task of generating summaries from textual documents is called text summarization. According to the way the summary is generated, text summarization can be categorized as extractive and abstractive summarization [6]. In the former type of summarization, the method only extracts sentences from the original text, while in the latter type of summarization, some additional processes are performed to the extracted sentences which makes the resulting summary contain different sentences from the text [7].

TextRank [8] is an extractive summarization method that uses graph to capture the relationship between sentences. and use this relationship to identify the importance of sentences. TextRank is an unsupervised method that has been used in many previous work on summarization, either as main or baseline method, because of its proven effectiveness [8]–[11]. Previously, many studies have also been conducted to enhance the performance of the TextRank algorithm for text summarization or keyword extraction, such as using word embedding [12]–[18], term frequency-inverse document frequency (TF-IDF) [19], [20], the combination of 1 gram, 2 gram, and Hidden Markov models [21], knowledge graph sentence embedding and K-means clustering [22], statistical and linguistic features for sentence weighting [23], variation of sentence similarity functions [24], and fine-tuning the hyperparameters [13].

In this work, we propose to use weighted word embedding using a variation of word embeddings, and TF-IDF weighting to enhance the effectiveness of the TextRank algorithm. Word embedding is used to obtain a better word representation that can capture semantic information, resulting in better sentence representation. Then, TF-IDF is used to weigh the generated word embedding to further improve sentence representation; this is based on our intuition that more important words, which are estimated using corpus statistics, should be valued more when generating the vector representation of sentences. TF-IDF is chosen because it has been shown to perform well for term weighting in various natural language processing tasks [19], [20], [25]–[27], and it also has been proven to significantly outperform the bag-of-words (BoW) technique [28]. The combination of word embedding and TF-IDF weighting components are then expected to improve the estimation of sentence relationships in the TextRank algorithm. We use two traditional word embedding models, i.e., Word2Vec [29] and FastText [30]), and one contextualized word embedding model, i.e., bidirectional encoder representations from transformers (BERT) [31] that has been pre-trained using a large-scale Indonesian corpus, referred to as Indonesian BERT (IndoBERT)-based model [32].

We divide some previous works on text summarization that are related to our work into three groups. First, the works that used word embedding in TextRank. Second, the works that used TF-IDF weighting in TextRank. Third, the works that used weighted word embedding. They are explained in more detail in the following three paragraphs. To the best of our knowledge, none of the previous works has incorporated weighted word embedding in TextRank summarization algorithm. In addition, none of them have investigated the use of contextualized word embedding from IndoBERT in TextRank for text summarization. Therefore, these become the research gaps that will be filled in this work.

Many attention has been given to the use of word embedding in TextRank algorithm for text summarization [12]–[16]. These works use different kinds of word embeddings, such as Global Vectors for word representation (GloVe) [12], [14], [17], Word2Vec [13], [14], FastText [14], and sentence-BERT (SBERT) [15], [16]. All of these works, however, did not use weighted word embedding, so the embedding is not weighed according to the collection statistics. In addition, they also did not investigate the contextual word embedding from IndoBERT, which becomes one of the words embedding studied in this work to be incorporated with TextRank. These things differentiate between our work and their work.

Using TextRank and TF-IDF to generate extractive summaries from documents has been explored by relatively a few studies [19], [20]. Zaware *et al.* [19] used TF-IDF as a vector representation of sentences to be inputted into the TextRank algorithm. More specifically, a sentence is represented using an n-dimensional vector space containing the TF-IDF scores of words. Different from them, Guan *et al.* [20] did not incorporate TF-IDF and TextRank, but they used them separately to extract keywords from the text and the comments given to the text, respectively. The resulting keywords were then used to score the sentences. In contrast to Zaware *et al.* [19] and Guan *et al.* [20] we use TF-IDF to weigh the word embedding to improve the vector representation of sentences.

There are relatively very few works that have investigated weighted word embedding for summarization [11], [33], but they did not use them with TextRank algorithm. Rani and Labiyal [11] used weighted word embedding, combined with a set of linguistic and statistical features to rank sentences in the clusters generated using K-means algorithm. You *et al.* [33] explored the use of IDF weight and Word2Vec embedding as one of the reconstruction mechanisms in their sequence-to-sequence summarization model. Our work is different to them as we incorporate the weighted word embedding into the TextRank summarization algorithm.

Our research questions in this work are as follows: i) How is the effectiveness of using word embedding from Word2Vec, FastText, and BERT in the TextRank algorithm for extracting text summaries in the Indonesian dataset? and ii) To what extent does the performance of the summarization systems differ when the word embeddings are weighed using TF-IDF? The rest of the paper is then organized as follows.

Section 2 describes the dataset, system framework, and evaluation method. Section 3 describes our experimental results together with the analysis. At last, section 4 concludes our study.

2. METHOD

2.1. Dataset

This work uses Liputan6 dataset [9], a large-scale Indonesian dataset for text summarization. The documents in this dataset come from news articles on the Indonesian online news portal Liputan6. It covers a variety of topics, such as: politics, business, and technology. This dataset was automatically built by Koto *et al.* [9], extracting the short description contained in the webpage metadata for each news article in the Liputan6 website, as the ground truth summary for the article. We use the Canonical set of the Liputan6 dataset, which is a complete version of this dataset, to obtain more comprehensive results in our experiment. Since the main summarization method used in this work is unsupervised, i.e., TextRank algorithm, we then only use the test split of the dataset which consists of 10,972 documents. The statistics of this dataset is presented in Table 1.

Table 1. Statistics of our dataset

Characteristic	Statistic
Total documents	10,972 documents
Average length of document (in sentences)	11.71 sentences
Average length of document (in words)	218.55 words
Average length of document sentence	18.67 words
Average length of ground truth summary (in sentences)	2.06 sentences
Average length of ground truth summary (in words)	26.20 words

It appears from the Table 1 that the documents in our dataset are short to moderate in length, which is around 12 sentences per document. The ground truth summary is also relatively short with 2 sentences on average, ranging from 1 to 5 sentences. For evaluation purposes, the length of document summaries generated by our summarization system will vary, following the length of ground truth summaries for the corresponding documents. For example, if a ground truth summary for a document has 3 sentences, then our summarization system will also produce a 3-sentence summary for that document.

One of the conveniences provided by Liputan6 dataset is that the tokenization has been done on each sentence of the news document, and each sentence has also been tokenized into words. This makes us easier to preprocess the data. A preprocessing that we performed on the dataset is only case-folding. Figure 1 illustrates an example of a document in the Liputan6 dataset that is used in this work (see the right column of the table). We can see that the documents in the Liputan6 dataset are in the form of a list of sentences, where each sentence is in the form of a list of words. It also does not include the document titles. The original document is presented in the left column of the table. Note that the English translation is not a part of the dataset, but we display it in the table to help readers understand the content of this document. This document example consists of nine sentences, and its ground truth summary consists of two sentences.

Original document (in Bahasa Indonesia):

Pemerintah Akan Mengembangkan Potensi Perikanan Budidaya

Liputan6.com, Jakarta: Berbagai kendala menghambat pendapatan negara dari sektor perikanan tangkap yang selama ini dijadikan andalan. Satu di antaranya adalah penangkapan ikan secara ilegal oleh kapal-kapal asing di wilayah perairan Indonesia. Walau berbagai cara telah dilakukan untuk menanggulangi nelayan asing ilegal, namun potensi perikanan tangkap tetap terganggu. Demikian diungkapkan Menteri Kelautan dan Perikanan Sarwono Kusumaatmaja, baru-baru ini, di Jakarta.
(5 kalimat sesudahnya tidak ditampilkan)

Ground truth summary:

Pemerintah bermaksud akan lebih mengandalkan sektor perikanan budidaya untuk meningkatkan pendapatan negara. Pasalnya, sektor perikanan tangkap yang selama ini dijadikan andalan sudah tidak optimal lagi.

English translation:

The Government Will Develop Aquaculture Potential

Liputan6.com, Jakarta: Various obstacles hinder state revenue from the capture fisheries sector which has been used as a mainstay. One of them is illegal fishing by foreign ships in Indonesian waters. Although various methods have been taken to tackle illegal foreign fishermen, the potential for capture fisheries remains disrupted. This was stated by the Minister of Maritime Affairs and Fisheries, Sarwono Kusumaatmaja, recently in Jakarta.
(the next 5 sentences are not displayed)

Ground truth summary:

The government intends to rely more on the aquaculture sector to increase state revenue. This is because the capture fisheries sector, which has been used as a mainstay, is no longer optimal.

Document inside Liputan6 dataset:

[
["Liputan6", ".", "com", ",", "Jakarta", ":", "Berbagai", "kendala", "menghambat", "pendapatan", "negara", "dari", "sektor", "perikanan", "tangkap", "yang", "selama", "ini", "dijadikan", "andalan", "."],
["Satu", "di", "antaranya", "adalah", "penangkapan", "ikan", "secara", "ilegal", "oleh", "kapal-kapal", "asing", "di", "wilayah", "perairan", "Indonesia", "."],
["Walau", "berbagai", "cara", "telah", "dilakukan", "untuk", "menanggulangi", "nelayan", "asing", "ilegal", ",", "namun", "potensi", "perikanan", "tangkap", "tetap", "terganggu", "."],
["Demikian", "diungkapkan", "Menteri", "Kelautan", "dan", "Perikanan", "Sarwono", "Kusumaatmaja", ",", ",", "baru-baru", "ini", ",", ",", "di", "Jakarta", "."],
(5 kalimat sesudahnya tidak ditampilkan)

Ground truth summary:

[
["Pemerintah", "bermaksud", "akan", "lebih", "mengandalkan", "sektor", "perikanan", "budidaya", "untuk", "meningkatkan", "pendapatan", "negara", "."],
["Pasalnya", ",", "sektor", "perikanan", "tangkap", "yang", "selama", "ini", "dijadikan", "andalan", "sudah", "tidak", "optimal", "lagi", "."]
]

Figure 1. A sample of text in Liputan6 dataset (right) and its original text in the Liputan6 website (left)

We also use another dataset to train the embedding models from which the word embeddings are extracted. A detailed explanation of the word embedding methods used in this work is explained in section 2.3.1. To train Word2Vec and FastText embedding models, we use a dump of the Indonesian Wikipedia on September 20th, 2022 (*idwiki-20220920-pages-articles.xml.bz2*) which consists of 3,309,595 pages (according to the page count statistic presented in the idwiki-20220920-site stats.sql.gz) or 135,678,726 words. For BERT embedding, we use a pertained language model BERT [32] that was trained on a large-scale Indonesian corpus, i.e., Indo4B dataset which consists of a variety of data, such as Wikipedia pages, news articles, and tweets, with a total of 3,581,301,476 words. This model was called IndoBERT in Wilie *et al.* [32].

2.2. System framework

Figure 2 illustrates the flow of the process in our summarization system. Initially, embedding models are learned using a large corpus. For Wod2Vec and FastText models, we train them using Indonesian Wikipedia dataset. For the BERT model, we use Indonesian BERT-based pre-trained models from previous work, i.e., IndoBERT [32].

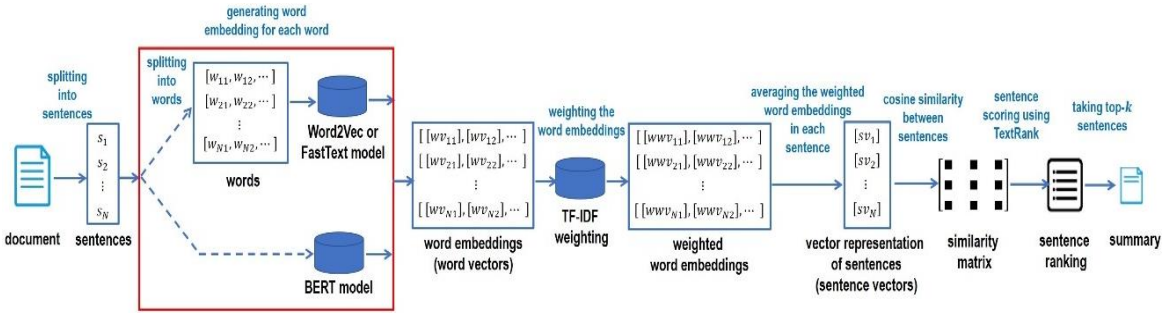

Figure 2. The framework of our enhanced TextRank summarization system

When a document to be summarized is inputted into our system, at first it needs to be split into sentences (*s*). Then, word embedding for each word in the sentences is generated from the embedding models. The red box in the figure highlights a slightly different process in generating word embedding using Word2Vec/FastText and BERT models. The dashed blue arrows inside the red box indicate conditional flow. When we want to generate word embedding using Word2Vec/FastText, then we follow the top dashed arrow. Otherwise, we follow the bottom dashed arrow when we want to generate BERT word embedding.

Because Word2Vec and FastText are traditional word embedding, there is only one static vector representation for each word. Therefore, each sentence needs to be split into words first. Then, given a word (w), the Word2Vec/FastText model will generate a word embedding or word vector (wv). This is different from BERT which is a contextualized word embedding, so it considers context from the right and left word sequences when computing a word embedding. As a result, the model will give different embedding for a word depending on its context, therefore, the word embedding for any word is not static. Since the model needs context, then we directly input a sentence into the model to produce word embedding for each word in the sentence.

Further, TF-IDF weighting is computed using our Liputan6 dataset to assign a weight for each word based on their occurrence statistics in the dataset. The TF-IDF scores are used to weigh the word embedding generated earlier and then produce the weighted word embedding or the weighted word vector (wwv). Next, sentence vectors (sv) are formed by taking the average weighted word vectors that compose the sentences. After sentence vectors are obtained, cosine similarity between any two sentences in the document is computed to build a sentence similarity matrix. TextRank summarization algorithm is then applied to score the sentences based on their importance among all other sentences in the document using graph-based method. To generate a summary, a top-*k* sentences with the highest scores will be extracted and then ordered according to the order of sentences in the document.

In general, our summarization system contains three main components. They are word embedding, TF-IDF weighting, and TextRank summarization algorithm. More detailed explanation about these components is given in the following subsections.

2.2.1. Word embedding

Word embedding concept was initially introduced by Bengio *et al.* [34] as a distributed representation of words, obtained after jointly training word embeddings with a model's parameter in a

neural network model, that can preserve the semantic and syntactic relationship between words. Word embedding represents a word as a vector learned from a large collection of data, in which words that have similar meanings (semantically similar) will have vectors that are close in the vector space. Word embedding is generally more effective than classic bag-of-words vector representation because of its ability to capture the semantic relationship between words. In addition, since word embedding has dense representation, then it also solves the sparsity problem in the classic representation, which makes the computation more effective and efficient. Therefore, most of the current research in text processing uses word embedding as the word representation [35]–[38]. There are three variants of word embedding that are explored in this work: Word2Vec, FastText, and BERT. They are detailed in the following.

a) Word2Vec

Mikolov *et al.* [29] introduced two neural architectures to learn Word2Vec word embeddings: the continuous skip-gram model and the continuous bag of words (CBOW) model. Both models learn vector representations of words that can capture semantic information as well as linguistic regularities and patterns. CBOW works on the task of predicting the current word $w(t)$ based on context words $w(t \pm i)$, where $i = 1,2,3, \dots, n$, with n denotes the window size. On the contrary, Skip-gram works on the task of predicting context words $w(t \pm i)$ based on current word $w(t)$. Window size is one of the hyperparameters in the Word2Vec algorithm, which can be set during the training process, that indicates the maximum distance between the current and predicted words within a sentence. The weights in the neural network learned during the training process will serve as the elements of word embedding. Mikolov *et al.* [29] found that the skip-gram model can capture semantic relationships better than the CBOW model. Therefore, the skip-gram model is chosen in this work to build the Word2Vec model.

We use a Python library *Gensim* to implement the Word2Vec algorithm and train the corresponding embedding model using Indonesian Wikipedia dataset. We use *Gensim*'s default value for the hyperparameter values to build the Word2Vec model, as follows: learning $rate = 0.025$, $epoch = 5$, and window $size = 5$. We train the Word2Vec model using Indonesian Wikipedia dataset because it is a large dataset that is publicly available. Using the same reason, there are also many previous researchers who also learned their Word2Vec models using Wikipedia dataset [39], [40].

b) FastText

FastText is an extension of Word2Vec embedding proposed by Bojanowski *et al.* [30] that takes into account the morphology of words (i.e., subwords). In the original Word2Vec embedding, every word has its word embedding that is generated without considering the similarity in the word morphology. So, it is possible that two morphologically similar words do not have similar word representations. This is considered a drawback of Word2Vec for a language that is rich in morphology [30]. FastText operates at a more granular level which is at the character *n*-gram level, while Word2Vec works at the word level. Character *n*-gram is a sequence of *n* characters within a given character window. In FastText, a word is represented by a sum of its character *n*-grams. Therefore, word embedding is obtained by summing up the vector representation of its character *n*-grams. For example, Indonesian word "*melihat*" (English translation: "see") with a window size of 3 for example, will be split up into seven subwords (character 3-gram): "<me", "mel", "eli", "lih", "iha", "hat", "at>", and the word embedding for that word is the sum of vectors for those seven subwords.

Using the above mechanism, FastText can learn representations for morphologically rich languages better. Words that have similar morphology (sharing many overlapping character *n*-grams) will be closer in vector space. In addition, it also enables rare words to be represented appropriately by taking the summation of embedding of its subwords. As a result, we can infer the embedding of unseen words that do not exist in the training corpus, which therefore helps to tackle the out of vocabulary (OOV) issue. Similar to Word2Vec implementation, we also use a Python library *Gensim* and Skip-gram architecture to implement the FastText algorithm and train the corresponding embedding model using Indonesian Wikipedia dataset.

c) Bidirectional encoder representations from transformers

BERT is a contextualized language model that was first introduced by Devlin *et al.* [31]. BERT architecture contains a stack of encoders from the transformer model [41], that learns the word context of a language in a bidirectional way. While Word2Vec and FastText belong to a traditional word embedding in which one word will always have one embedding, no matter what words exist before or after it, BERT belongs to the contextualized word embedding in which the word embedding is assigned by looking at the context around that word. Therefore, it is designed to better understand the contextual relationship between words. BERT model can be fine-tuned to directly solve the downstream tasks. In this work, however, BERT is only used in a feature-based approach to extract word embedding to be incorporated with the TextRank algorithm. The performance will then be compared against Word2Vec and FastText models.

Recently, BERT was pre-trained on a large-scale of Indonesian corpus consisting of around 4 billion words, referred to as IndoBERT [32]. IndoBERT has some variants that differ in the model architecture, such

as the number of encoder layers, the number of hidden layers, and the number of attention heads. Two main variants of IndoBERT that will be investigated in this work are IndoBERT$_{base}$ and IndoBERT$_{large}$. Table 2 highlights the differences in the architecture between IndoBERT$_{base}$ and IndoBERT$_{large}$.

Table 2. The architecture of IndoBERT$_{base}$ and IndoBERT$_{large}$ models

IndoBERT Model	#Layers	Hidden size	#Attention heads	Vocab size	#Parameters
IndoBERT$_{base}$	12	768	12	30,522	124.5M
IndoBERT$_{large}$	24	1,024	16	30,522	335.2M

It appears from the table that IndoBERT$_{large}$ has twice more encoders and almost three times more neural network parameters than IndoBERT$_{base}$. To examine to what extent this difference will affect the effectiveness of the resulting word embedding, we compare the results of our summarization system using contextualized embeddings from these two architectures. To use pre-trained IndoBERT models, we use library *transformer* in the hugging face website that provides application programming interface (APIs) and tools for downloading and training some state-of-the-art pre-trained language models.

2.2.2. TF-IDF weighting

TF-IDF is an attempt to give weight to words contained in a text based on their occurrence in the collection. In this work, TF-IDF score is used to weigh the word embeddings (word vectors). In general, TF-IDF estimates how important words are in a document, while also considering their importance in the collection. The formula to compute TF and IDF scores are (1) and (2),

$$tf(t,d) = \frac{f_{(t,d)}}{\sum_{\hat{t} \in d} f_{(\hat{t},d)}} \quad (1)$$

$$idf(t) = \frac{|C|}{|d \in C : t \in d|} \quad (2)$$

where $f_{t,d}$ represents the term frequency (TF) of a word t in a document d, and $\sum_{\hat{t} \in d} f_{(\hat{t},d)}$ indicates the document length, i.e., the total terms in the document d. The higher the number of occurrences of a term in a document, the more important that word in the document. Then, $idf(t)$ represents the inverse document frequency (IDF) of a word t in the collection C, where $|C|$ is the number of documents in the collection and $|d \in C : t \in d|$ is the number of documents in the collection that contain a word t. The more documents in a collection containing a particular word, the more general the word is. To sum up, words that often appear in a document, but do not often appear in the corpus will be assigned a high TF-IDF weight, indicating that they are important words.

2.2.3. TextRank

TextRank is a graph-based ranking algorithm for scoring the text [8], where the text unit can be specified, such as keywords or sentences. TextRank is an extension of PageRank [42] which was originally aimed for webpage ranking. In this paper, TextRank is used for sentence ranking purposes. The relationship between sentences in the document is a factor that contributes to the score given by TexRank to each sentence. To calculate the relationship between sentences, we use cosine similarity, following Barrios *et al.* [24]. The resulting pairwise cosine similarity scores between sentences are then used to construct a similarity matrix that serves as a basis to form a weighted graph of sentences.

In the weighted graph representation, a vertex represents a sentence, and an edge between two vertexes represents the relationship/connection between two sentences. The similarity score is used as the weight of the edge between those two sentences. Two sentences that are more similar will have a higher weight on the edge connecting them. Based on this graph representation, we then apply TextRank algorithm to score each sentence in the document using (3),

$$S(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{v_k \in Out(V_j)} w_{jk}} * S(V_j) \quad (3)$$

where V_i is the i-th vertex and $S(V_i)$ denotes the score for the i-th vertex, which basically represents the sentence score. $In(V_i)$ and $Out(V_j)$ respectively denotes the set of vertexes that go into V_i and the set of vertexes that go out from V_j. d is a damping factor, which is the probability of randomly moving from one vertex to another vertex in the graph. w_{ji} and w_{jk} respectively denotes the weight of edge between sentences j and i, and the weight of edge between sentences j and k.

TextRank is an iterative algorithm that will stop the iteration when convergence is achieved. It is when there is a sentence in the document whose the difference between its scores in two successive iterations is already very small so we can assume that in the next iterations the scores will not significantly change anymore. When convergence is achieved, all sentences in a document have final scores. We will then take *N* sentences with the highest scores as a summary. Since the length of the summary in our ground truth dataset is varied, then *N* is adjusted with the actual length of the summary for a document. In our implementation, we use Python library *sklearn* to compute the cosine similarity between sentences. Then, the Python library *NetworkX* is used to implement the TextRank algorithm. The damping factor is set to 0.85, following the implementation of original PageRank [42] and TextRank [8]. For an original TextRank that is used as a baseline method in our experiments, the word representation uses a vector of bag-of-words as in the original paper [8]. For our summarization methods, we use word embedding from Word2Vec, FastText, and IndoBERT as word representations.

2.3. Evaluation method

The evaluation of our experiment results is performed using recall-oriented understudy for gisting evaluation (ROUGE) metric [43], which is a common metric for extractive text summarization. In general, ROUGE calculates the word overlap between automatic summaries and ground truth summaries. The types of ROUGE to be used are ROUGE-N and ROUGE-L. While ROUGE-N counts the number of n-grams overlapping between two summaries, ROUGE-L counts the LCS (longest common subsequence) between two summaries. We choose F-1 score as the type of measurement for our ROUGE scores calculation since it considers both Precision and Recall. A Python library *rouge_score* is used to calculate ROUGE scores for our summaries.

3. RESULTS AND DISCUSSION

3.1. The effectiveness of enhanced TextRank using (unweighted) word embedding

Table 3 describes the performance of our TextRank-based summarization systems using (unweighted) word embedding from Word2Vec, FastText, and IndoBERT. These systems are compared to the original TextRank method as the baseline. We can see that the ROUGE scores of all systems using word embedding are higher than the baseline system. Here, the use of word embedding is shown to significantly increase the performance of the original TextRank algorithm.

Table 3. Performance comparison between systems using (unweighted) word embedding

Summarization System	ROUGE-1	ROUGE-2	ROUGE-L
TextRank	0.3479	0.2339	0.3302
TextRank+Word2Vec	0.3822*	0.2693*	0.3646*
TextRank+FastText	0.3776*	0.2644*	0.3598*
TextRank+IndoBERT$_{base}$	0.3929$^{*+\times}$	0.2837$^{*+\times}$	0.3768$^{*+\times}$
TextRank+IndoBERT$_{large}$	0.3940$^{*+\times}$	0.2854$^{*+\times}$	0.3779$^{*+\times}$

Symbols *, +, ×, and ÷ denote significant differences against TextRank, TextRank + Word2Vec, TextRank + FastText, and TextRank + IndoBERTbase methods, respectively, according to the paired t-test (p <0.05).

It also appears from the table that the systems using IndoBERT are significantly more effective than those using Word2Vec and FastText. This confirms one of our contributions in this work which is using the contextualized word embedding BERT in the TextRank algorithm. The advantage possessed by BERT occurs because its word embeddings are generated by looking at the context of the surrounding words (this is different from Word2Vec and FastText models that adopt static word representation as it does not consider the context when generating word embedding). As a result, the word embedding from BERT is more accurate to capture semantic relationships between words. A better word representation then results in a better sentence representation, which further contributes to a better estimate of sentence importance by the TextRank method. This explains the success of BERT embedding in enhancing the performance of the TextRank method the most.

TextRank using IndoBERTlarge gains slightly higher ROUGE scores than that using IndoBERTbase, and this result is consistent with the findings reported in previous work [44]. The statistical test, however, shows that this difference is not significant. This slight superiority of IndoBERTlarge over IndoBERTbase is affected by the higher number of layers as well as parameters used in its neural model, which makes it slightly better at capturing the meanings of words.

3.2. The effectiveness of enhanced TextRank using weighted word embedding

Table 4 shows the performance of the enhanced TextRank methods using weighted word embedding. Compared to the results using unweighted word embedding presented in Table 3 earlier, it is clear that the application of TF-IDF weighting to the word embedding gives significant performance increases to all summarization systems. This increase can occur because by using weighted word embedding, the importance of words in the collection is considered when making the sentence representation. Consequently, it improves the generated sentence vectors which then results in enhancing the performance of TextRank method to estimate the importance of sentences.

Table 4. Performance comparison between systems using weighted word embedding

Summarization System	ROUGE-1	ROUGE-2	ROUGE-L
TextRank	0.3479	0.2339	0.3302
TextRank+Word2Vec+TF-IDF	$0.4082^{\ddagger*}$	$0.3041^{\ddagger*}$	$0.3926^{\ddagger*}$
TextRank+FastText+TF-IDF	$0.4042^{\ddagger*}$	$0.2982^{\ddagger*}$	$0.3879^{\ddagger*}$
TextRank+IndoBERT$_{base}$+TF-IDF	$0.4039^{\ddagger*}$	$0.2974^{\ddagger*}$	$0.3880^{\ddagger*}$
TextRank+IndoBERT$_{large}$+TF-IDF	$0.4044^{\ddagger*}$	$0.2982^{\ddagger*}$	$0.3884^{\ddagger*}$

Symbol ‡ illustrates the significant difference against the corresponding methods without using TF-IDF weighting, as displayed in Table 3. Then, symbols *, +, ×, and ÷ denote significant differences against TextRank, TextRank + Word2Vec + TF-IDF, TextRank + FastText + TF-IDF, and TextRank + IndoBERT-based-methods + TF-IDF, respectively, according to the paired t-test (p <0.05)

The increased levels of systems performance over those without using weighted word embedding are 2.64% to 7.04% in ROUGE-1, 4.48% to 12.92% in ROUGE-2, and 2.78% to 7.81% in ROUGE-L. Systems that are benefited the most from the TF-IDF weighting are those using the Word2Vec and FastText models, which respectively achieve 12.92% and 12.78% increases in ROUGE-2. For systems using IndoBERT$_{base}$ and IndoBERT$_{large}$ models, they only gain 4.83% and 4.48% increases in ROUGE-2, respectively. Here, we can see that the performance increases obtained by the systems using Word2Vec and FastText models are almost three times higher compared to those using the BERT models. We analyze that this could happen because the TF-IDF weighting can help to reduce the limitation of static word embeddings generated by Word2Vec and FastText, which do not see the contextual meaning of a word. Therefore, the TF-IDF weighting can give extra information on the importance of words in the collection to the word representation. In addition, the way TF-IDF works is almost the same as word embedding from Word2Vec and FastText, where each word has only one TF-IDF weight no matter what words occur before or after it as it does not consider the context. Consequently, it is more effective for Word2Vec and FastText, rather than BERT.

3.3. More about the summary results

We analyze some examples of the summaries generated using our systems and compare them with ground truth summaries. Table 5 shows an example of the results of our summaries generated for a document with id "16,379" in our dataset using TextRank and (unweighted/weighted) word embedding based on Word2Vec, FastText, and BERT. The ground truth summary as well as the baseline summary (TextRank) for this document are also displayed in the table for comparison. The text printed in boldface indicates the text that exists in the ground truth summary. The more the text in an automatically generated summary is bolded, the better the summary is, since it indicates that it is more similar to the ground truth summary.

It appears that the summary generated using the baseline method contains only one sentence (out of two sentences) from the ground truth summary. The result is the same as those of TextRank+Word2Vec and TextRank+FastText systems. In this case, the use of unweighted word embedding from Word2Vec and FastText is unable to improve the baseline method, i.e., TextRank.

However, after adding the TF-IDF weighting to the word embedding from TextRank and Word2Vec, it succeeds to enhance the TextRank method, which enables the summaries generated by these systems to be the same as the ground truth summary. This example demonstrates that weighted word embedding is effective for Word2Vec and FastText, as it can improve the performance of unweighted word embedding. Meanwhile, the use of the IndoBERTbase and IndoBERTlarge word embeddings, even without TF-IDF weighting, has succeeded to produce summaries that are the same as the ground truth summary. Further using the weighted word embedding does not change the summary results for IndoBERT-based systems. This confirms our results in the earlier subsection that since IndoBERT considers the context to produce word embedding, then using the original word embedding can already capture the semantics of words well. Therefore, the effect of TF-IDF weighting for this method is smaller compared to that for static word embeddings (Word2Vec and FastText).

Table 5. Example of summaries for a document id "16379"

System	Summaries
Ground Truth	*Rakernas juga akan meminta penjelasan Fraksi PDI-P di DPR tentang proses Memorandum I, II, hingga menjelang SI MPR. Rakernas juga akan menanyakan kesiapan F-PDIP MPR menghadapi sidang istimewa*. (English translation: The National Working Meeting will also ask for an explanation from the PDI-P faction in the People's Representative Council regarding the process of Memorandums I, II, up to the time before the people's consultative assembly (PCA) Special Session. The National Working Meeting will also ask about the PCA's F-PDIP readiness for a special session.)
TextRank (baseline)	*Di antaranya, membicarakan soal situasi politik terkini, termasuk pergelaran Sidang Istimewa MPR, 1 Agustus mendatang. Rakernas juga akan meminta penjelasan Fraksi PDI-P di DPR tentang proses Memorandum I, II, hingga menjelang SI MPR*. (ROUGE-1=0.6875, ROUGE-2=0.6129, ROUGE-L=0.6875) (English translation: Among other things, discussing the current political situation, including the holding of the MPR Special Session, 1 August. The National Working Meeting will also ask for an explanation from the PDI-P faction in the People's Representative Council regarding the process of Memorandums I, II, up to the time before the PCA Special Session.)
TextRank+Word2Vec	same as the summary of TexRank above
TextRank+FastText	same as the summary of TexRank above
TextRank+ $IndoBERT_{base}$	*Rakernas juga akan meminta penjelasan Fraksi PDI-P di DPR tentang proses Memorandum I, II, hingga menjelang SI MPR. Rakernas juga akan menanyakan kesiapan F-PDIP MPR menghadapi sidang istimewa*. (ROUGE-1=1, ROUGE-2=1, ROUGE-L=1) (English translation: The National Working Meeting will also ask for an explanation from the PDI-P faction in the People's Representative Council regarding the process of Memorandums I, II, up to the time before the PCA Special Session. The National Working Meeting will also ask about the PCA's F-PDIP readiness for a special session.)
TextRank+ $IndoBERT_{large}$	same as the summary of TexRank+$IndoBERT_{base}$ above
TextRank+Word2Vec+ TF-IDF	same as the summary of TexRank+$IndoBERT_{base}$ above
TextRank+FastText+ TF-IDF	same as the summary of TexRank+$IndoBERT_{base}$ above
TextRank+ $IndoBERT_{base}$+TF-IDF	same as the summary of TexRank+$IndoBERT_{base}$ above
TextRank+ $IndoBERT_{large}$+TF-IDF	same as the summary of TexRank+$IndoBERT_{base}$ above

4. CONCLUSION

We propose to use weighted word embedding to enhance TextRank algorithm for text summarization on Indonesian news dataset. A variation of word embeddings, such as traditional word embedding (Word2Vec & FastText) and contextualized word embedding BERT, combined with TF-IDF weighting are exploited in our methods to be incorporated with the TextRank. The effect of the weighted word embedding (as well as unweighted word embedding) on the performance of TextRank summarization method are examined in this work.

Our experimental results show that the use of (unweighted) word embedding improves the performance of the TextRank method by up to 13.25% in ROUGE-1 and up to 22.02% in ROUGE-2. The system using word embedding from BERT is shown to achieve the highest performance increase. BERT-based summarization systems gain 5.98% and 7.94% higher ROUGE-2 scores as compared to the systems using Word2Vec and FastText, respectively. This indicates that BERT word embedding, which is generated by considering the context of the surrounding words, produces better word representation than Word2Vec and FastText, which further results in better estimation of sentence importance in TextRank. When each word embedding is weighed using TF-IDF score, we found that the performance for all systems using unweighted word embedding significantly improve. However, BERT-based systems only gain a little improvement (with 2.64% to 4.83% increase), while the biggest improvement is achieved by the systems using static word embeddings, i.e., Word2Vec (with 6.80% to 12.92% increase) and FastText (with 7.04% to 12.78% increase). Overall, our systems using weighted word embedding can outperform the original TextRank method by up to 17.33% in ROUGE-1 and 30.01% in ROUGE-2. This shows that our proposed methods succeeded to give a significant improvement over the original TextRank method.

ACKNOWLEDGEMENT

This research was funded by the Directorate of Research and Development, Universitas Indonesia, under Hibah PUTI Q2 2022 (Grant No. NKB-571/UN2.RST/HKP.05.00/2022).

REFERENCES

[1] S. Kemp, "Digital 2022: Indonesia," *DataReportal*, 2022. https://datareportal.com/reports/digital-2022-indonesia (accessed Jul. 31, 2022).

[2] R. D. Lestari, "Shifting journalistic ethics in the internet age, case study: Violation of journalistic ethics in journalistic products and journalist behavior in online media," *Komunikator*, vol. 11, no. 2, 2019, doi: 10.18196/jkm.112027.

[3] N. Kurniasih, "Reading habit in digital era: Indonesian people do not like reading, is it true?," in *INA-Rxiv Papers*, 2017, pp. 1–4.

[4] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, Apr. 1958, doi: 10.1147/rd.22.0159.

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

[6] A. Pai, "Text summarizer using abstractive and extractive method," *International Journal of Engineering Research and Technology (IJERT)*, vol. 3, no. 5, pp. 971–975, 2014.

[7] A. Nenkova, "Automatic summarization," *Foundations and Trends®in Information Retrieval*, vol. 5, no. 2, pp. 103–233, 2011, doi: 10.1561/1500000015.

[8] R. Mihalcea and P. Tarau, "TextRank: bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Jul. 2004, pp. 404–411.

[9] F. Koto, J. H. Lau, and T. Baldwin, "Liputan6: A large-scale Indonesian dataset for text summarization," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 2020, pp. 598–608.

[10] B. Balcerzak, W. Jaworski, and A. Wierzbicki, "Application of TextRank algorithm for credibility assessment," in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 2014, vol. 1, pp. 451–454, doi: 10.1109/WI-IAT.2014.70.

[11] R. Rani and D. K. Lobiyal, "A weighted word embedding based approach for extractive text summarization," *Expert Systems with Applications*, vol. 186, 2021, doi: 10.1016/j.eswa.2021.115867.

[12] S. Sumana, "Towards automatically generating release notes using extractive summarization technique," in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, Jul. 2021, pp. 241–248, doi: 10.18293/SEKE2021-119.

[13] D. Jain, M. D. Borah, and A. Biswas, "Fine-tuning Textrank for legal document summarization: a Bayesian optimization based approach," in *ACM International Conference Proceeding Series*, 2020, pp. 41–48, doi: 10.1145/3441501.3441502.

[14] T. T. Hailu, J. Yu, and T. G. Fantaye, "A framework for word embedding based automatic text summarization and evaluation," *Information*, vol. 11, no. 2, Jan. 2020, doi: 10.3390/info11020078.

[15] A. Kazemi, V. Pérez-Rosas, and R. Mihalcea, "Biased TextRank: Unsupervised graph-based content extraction," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 1642–1652, doi: 10.18653/v1/2020.coling-main.144.

[16] S. Gupta, A. Sharaff, and N. K. Nagwani, "Biomedical text summarization: a graph-based ranking approach," in *Advances in Intelligent Systems and Computing*, Springer Singapore, 2022, pp. 147–156, doi: 10.1007/978-981-16-2008-9_14.

[17] U. Barman, V. Barman, M. Rahman, and N. K. Choudhury, "Graph based extractive news articles summarization approach leveraging static word embeddings," in *2021 International Conference on Computational Performance Evaluation (ComPE)*, 2021, pp. 8–11, doi: 10.1109/ComPE53109.2021.9752056.

[18] X. Zuo, S. Zhang, and J. Xia, "The enhancement of TextRank algorithm by using Word2Vec and its application on topic extraction," *Journal of Physics: Conference Series*, vol. 887, no. 1, 2017, doi: 10.1088/1742-6596/887/1/012028.

[19] S. Zaware, D. Patadiya, A. Gaikwad, S. Gulhane, and A. Thakare, "Text summarization using TF-IDF and TextRank algorithm," in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, Jun. 2021, pp. 1399–1407, doi: 10.1109/ICOEI51242.2021.9453071.

[20] X. Guan, Y. Li, Q. Zeng, and C. Zhou, "An automatic text summary extraction method based on improved TextRank and TF-IDF," *IOP Conference Series: Materials Science and Engineering*, vol. 563, no. 4, Jul. 2019, doi: 10.1088/1757-899X/563/4/042015.

[21] X. Zheng, T. Zhou, Y. Wang, and S. Li, "An improved TextRank-based method for chinese text summarization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13339, Springer International Publishing, 2022, pp. 140–149, doi: 10.1007/978-3-031-06788-4_12.

[22] T. Tang, T. Yuan, X. Tang, and D. Chen, "Incorporating external knowledge into unsupervised graph model for document summarization," *Electronics*, vol. 9, no. 9, Sep. 2020, doi: 10.3390/electronics9091520.

[23] S. Yu, J. Su, P. Li, and H. Wang, "Towards high performance text mining: A TextRank-based method for automatic text summarization," *International Journal of Grid and High Performance Computing*, vol. 8, no. 2, pp. 58–75, Apr. 2016, doi: 10.4018/IJGHPC.2016040104.

[24] F. Barrios, F. López, L. Argerich, and R. Wachenchauzer, "Variations of the similarity function of TextRank for automated summarization," *Prepr. arXiv.1602.03606*, Feb. 2016.

[25] K. Tamara and N. Milicevic, "Comparing sentiment analysis and document representation methods of amazon reviews," in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, Sep. 2018, pp. 000283–000286, doi: 10.1109/SISY.2018.8524814.

[26] D. E. Cahyani and I. Patasik, "Performance comparison of TF-IDF and Word2Vec models for emotion text classification," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 10, no. 5, pp. 2780–2788, 2021, doi: 10.11591/eei.v10i5.3157.

[27] V. Kalra, I. Kashyap, and H. Kaur, "Improving document classification using domain-specific vocabulary: Hybridization of deep learning approach with TFIDF," *International Journal of Information Technology*, vol. 14, no. 5, pp. 2451–2457, 2022, doi: 10.1007/s41870-022-00889-x.

[28] S. Akuma, T. Lubem, and I. T. Adom, "Comparing bag of words and TF-IDF with different models for hate speech detection from live tweets," *International Journal of Information Technology*, vol. 14, no. 7, pp. 3629–3635, 2022, doi: 10.1007/s41870-022-01096-4.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, Jan. 2013.

[30] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *Prepr. arXiv.1810.04805*, Oct. 2018.

[32] B. Wilie *et al.*, "IndoNLU: benchmark and resources for evaluating Indonesian natural language understanding," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 2020, pp. 843–857.
[33] J. You, C. Hu, H. Kamigaito, H. Takamura, and M. Okumura, "Abstractive document summarization with word embedding reconstruction," in *Proceedings of the Conference Recent Advances in Natural Language Processing - Deep Learning for Natural Language Processing Methods and Applications*, 2021, pp. 1586–1596, doi: 10.26615/978-954-452-072-4_178.
[34] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "A neural probabilistic language models," in *Innovations in Machine Learning*, Berlin/Heidelberg: Springer-Verlag, 2006, pp. 137–186, doi: 10.1007/3-540-33486-6_6.
[35] T. V Rampisela and E. Yulianti, "Semantic-based query expansion for academic expert finding," in *2020 International Conference on Asian Language Processing (IALP)*, 2020, pp. 34–39, doi: 10.1109/IALP51396.2020.9310492.
[36] E. Yulianti, A. Kurnia, M. Adriani, and Y. S. Duto, "Normalisation of Indonesian-English code-mixed text and its effect on emotion classification," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 11, pp. 674–685, 2021, doi: 10.14569/IJACSA.2021.0121177.
[37] E. Yulianti, R.-C. Chen, F. Scholer, and M. Sanderson, "Using semantic and context features for answer summary extraction," in *Proceedings of the 21st Australasian Document Computing Symposium*, 2016, pp. 81–84, doi: 10.1145/3015022.3015031.
[38] T. V Rampisela and E. Yulianti, "Academic expert finding in Indonesia using word embedding and document embedding: A case study of Fasilkom UI," in *2020 8th International Conference on Information and Communication Technology (ICoICT)*, Jun. 2020, pp. 1–6, doi: 10.1109/ICoICT49345.2020.9166249.
[39] D. Jatnika, M. A. Bijaksana, and A. A. Suryani, "Word2Vec model analysis for semantic similarities in English words," *Procedia Computer Science*, vol. 157, pp. 160–167, 2019, doi: 10.1016/j.procs.2019.08.153.
[40] M. Al-Hajj and M. Jarrar, "LU-BZU at SemEval-2021 task 2: Word2Vec and Lemma2Vec performance in Arabic word-in-context disambiguation," in *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, 2021, pp. 748–755, doi: 10.18653/v1/2021.semeval-1.99.
[41] A. Vaswani *et al.*, "Attention is all you need," *Prepr. arXiv.1706.03762*, Jun. 2017.
[42] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, Apr. 1998, doi: 10.1016/S0169-7552(98)00110-X.
[43] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Association for Computational Linguistics*, 2004, pp. 74–81.
[44] N. Rachmawati and E. Yulianti, "Transfer learning for closed domain question answering in COVID-19," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 12, pp. 277–285, 2022, doi: 10.14569/IJACSA.2022.0131234.

BIOGRAPHIES OF AUTHORS

Evi Yulianti is a lecturer and researcher at the Faculty of Computer Science, Universitas Indonesia. She received the B.Comp.Sc. degree from the Universitas Indonesia in 2010, the dual M.Comp.Sc. degree from Universitas Indonesia and Royal Melbourne Institute of Technology University in 2013, and the Ph.D. degree from Royal Melbourne Institute of Technology University in 2018. Her research interests include information retrieval and natural language processing. She can be contacted at email evi.y@cs.ui.ac.id.

Nicholas Pangestu received his bachelor's degree in computer science from the Universitas Indonesia in 2023. His research interests are related to machine learning and natural language processing. He can be contacted at email pangestu.nicholas@gmail.com.

Meganingrum Arista Jiwanggi is a lecturer and researcher at Faculty of Computer Science, Universitas Indonesia. She received her bachelor's degree in computer science from the Universitas Indonesia in 2012 and a dual master of computer science degree from the Universitas Indonesia and Royal Melbourne Institute of Technology University in 2014. Her research interests include natural language processing and data science. She can be contacted at email meganingrum@cs.ui.ac.id.