Plant-Disease Relation Model through BERT-BiLSTM-CRF Approach

Slamet Riyanto[1], Imas Sukaesih Sitanggang[2], Taufik Djatna[3], Tika Dewi Atikah[4]
[1,2]Department of Computer Science, IPB University, Indonesia
[1]Research Center for Data and Information Science, National Research and Innovation Agency, Indonesia
[3]Department of Agro-Industrial Technology, IPB University, Indonesia
[4]Research Center for Ecology and Ethnobiology, National Research and Innovation Agency, Indonesia

Article Info

Article history:

Received Oct 14, 2023
Revised Feb 25, 2024
Accepted Mar 11, 2024

Keywords:

BERT
Bidirectional Long Short-Term Memory
Conditional Random Field
Relation Extraction
Plant-Disease Relation

ABSTRACT

Plant Disease Relations (PDR) is one of the Information Extraction (IE) subtasks that reveals the relationship between plant entities and diseases that appear together in a sentence. Previous studies have proposed methods for detecting the extraction of relationships between plant diseases (PDR). Previous research has proposed a Short Dependency Path-Convolutional Neural Network (SDP-CNN) method to predict relationships. However, the proposed method has limitations when faced with long and complex sentences. To overcome these limitations, this study proposes the BERT-BiLSTM-CRF method to improve the model performance in detecting PDR. First, the data is processed into the BERT Encoder layer after the tokenization process. After the BERT Encoder calculates the hidden information, the next step is to enter the linear layer to obtain word embedding. Calculation results in the bilinear layer are forwarded to the softmax layer to predict the relationship of each pair. Computation results in the softmax layer are sent to the BiLSTM layer. Finally, the CRF layer is entered to improve the prediction process. An 80:20 ratio for training and testing data was used to build the model using the same parameter values over ten attempts. GridSearch hyperparameter tuning is also involved in improving model performance. Experimental results show that the architecture proposed in this research can increase the F1 score by 0.790, which proved to be higher than SDP-CNN with a micro F1 score of 0.764. The problem of predicting PDR was overcome by the BERT-BILSTM-CRF method. The issue of forecasting PDR was resolved using the BERT-BILSTM-CRF approach.

Corresponding Author:

Slamet Riyanto
Department of Computer Science,
IPB University,
Jl. Meranti Wing 20 Level 5 Kampus IPB Darmaga 16680
Email: slametriyanto@apps.ipb.ac.id, slam011@brin.go.id

1. INTRODUCTION

Relation extraction is the task of natural language processing (NLP) to extract semantic relationships between entities from unstructured texts [1]. The importance of disclosing information on plant and disease relationships is to evaluate the usefulness of the plant disease corpus [2]. Various statistical machine-learning techniques have effectively tackled the relation extraction challenge because relation extraction may be transformed into a classification problem [3], [4].

Research on Plant-Disease Relation (PDR) was conducted by [2], who used the SDP-CNN method to predict multiple classes. The dataset used by Kim has unbalanced classes, i.e., 583 Negative (Neg), 507 Treatment of Disease (ToD), 183 Cause of Disease (CoD), and 34 Associations (Ass), bringing a total of 1307 data. The SDP technique relies heavily on part-of-speech tagging (POS tagging) to label each word and find the shortest dependencies between words in a sentence. The dependency relationship between words in a sentence is explained through the relation of subject, predicate, and object. In Natural Language Processing (NLP), the relationship between the constituents of a sentence is usually tree-like: words, phrases, and clauses form sentences hierarchically, and the dependencies between the different branches induce syntactic structure [5].

On the other hand, experimental results using the proposed method show that the model in this research obtains more satisfactory results than previous research. In addition, the latest LM included in the experiment also showed unsatisfactory performance. These results indicate that PDR research is significant and cannot be handled by the latest LM. This study shows a benchmark dataset from [2] to explain the distant supervision method on plant-disease relations.

Based on the literature that has been studied by [6]–[8], the techniques and methods used by [2] have limitations, namely: 1) focus on local features, 2) do not take context into account, 3) less effective for complex English, 4) the shortest path finding algorithm has high computational complexity, mainly when applied to long sentences. Referring to the limitations of previous research, a strategy is needed to extract relations between entities that still consider contextual sentences and are compatible with complex English structures.

NLP is a broad area for linguistic analysis of text using segmentation, tokenization, POS tagging, and syntactic parsing [9]. This study uses the intra-sentence approach in NLP terms. As is common knowledge, the Relation Extraction (RE) task consists of intra-sentence and inter-sentence. This study uses the RE sub-task to predict a single relation in the sentence. Hierarchically, text-based IE tasks are divided into four sub-tasks, namely, Named Entity Recognition (NER), Relation Extraction (RE), Event Extraction (EE), and Co-reference Resolution (CR) [10]. The RE function retrieves an event containing both a trigger and an argument, where triggers are verbs signaling events within the text [11]. This study uses BERT to train computers to understand the grammatical structures of LM tasks. LM is a basic NLP subtask often used to provide probability distributions of texts based on known text information [5].

The LM approach teaches computers to understand and produce linguistically meaningful texts. Therefore, LM is needed for NER, RE, EE and CR tasks. Several LMs currently available are Generative Pre-trained Transformer 3.5 (GPT-3.5), Google Bard, Bing Artificial Intelligence (Bing AI), and BERT. Some of these LMs have proven unable to solve plant-disease relation problems. This study conducted experiments on the latest LM, including ChatGPT 3.5, Bard, and Bing AI. The test data consisted of 100 randomly selected records. Each LM was treated in the same manner. At first, we provide input in the form of 4 predefined classes. Subsequently, unlabeled data is entered into the chat column, along with questions to predict the four classes.

The prediction results match the Gold Standard Corpus (GSC) data from previous research, available at http://gcancer.org/pdr (accessed on January 21, 2022. The test data used has multiple classes, and the correct prediction was given a score of 1, while the incorrect prediction was assigned a score of 0. The comparison of current LM performance for predicting PDR datasets still needs to be higher. Table 1 shows the percentage of expected results for each LM. Model performance is relatively similar, with an average accuracy of 40%. The level of similarity in predicting the same data obtains an accuracy score of 44%. It was observed that the proposed model is still important. Moreover, the results of these predictions are used as guidelines in the health and biomedical fields, with direct relevance to human life. Table 1 shows the failure of the three LMs, which refers to the result of predictions falling outside the four predefined classes and failing to understand the question.

Table 1. Comparative performance of language modelling

Metrics	ChatGPT	Google Bard	BingAI
Accuracy	41%	40%	41%
Similarity	44%	44%	44%
Failure	7%	12%	4%

This study trains all language models by mentioning four classes: Treatment of Disease, Case of Disease, Association, and Negative. After training, test data is input into the chat column, and orders are given to predict the class mentioned in the form of a question. The prediction results were matched with the gold standard corpus. If the prediction is wrong, a score of 0 will be given; if the prediction is correct, a score of 1 will be given to all language models. Score recording is done manually using the Spreadsheet application.

This study proposes Natural Language Processing (NLP) and Deep Learning (DL) to improve accuracy in predicting PDR based on previous studies. The contributions of this research are: 1) build an architecture of plant-disease relation, 2) develop an algorithm based on BERT-BiLSTM-CRF for predict PDR, 3) evaluate the performance of the BERT-BiLSTM-CRF plant-disease relation model, and 4) comparative analysis of latest LM in the classification of PDR. The rest of this paper is organised as follows. Section 2 discusses the dataset and the methodology used to undertake the research. The results and discussion follow this in Section 3. Finally, in Section 4, the conclusion and future works are presented.

2. PROPOSED RESEARCH METHOD

This section describes the stages of building a model to predict PDR using the BERT-BiLSTM-CRF algorithm with the oversampling method. The detailed steps are as follows: Source Data and Input Data, Preprocessing, Feature Extraction, Training Dataset, Relation Extraction Model, Calculation of Model Performance, and Confusion Matrix. In general, the research process is shown in Figure 1. The entire research experiment process was carried out in a working environment with the specifications: MacOS Monterey, 16GB RAM, 500GB HDD, 2.2 GHz Quad-Core Intel Core i7 Processor, and 1536 MB Intel Iris Pro Graphic to make the model. Meanwhile, the Hyperparameter Tuning process uses Google Colab Premium with specifications of 200 computing units, 12GB NVidia V100 GPU, and 225GB disk.

2.1. Source Data and Input Data

The source data used in this study was obtained from the gold standard corpora of PDR developed by [2]. The source data is used in the dataset training stage to establish a plant-disease relation classifier that labels relationships between entities. The BERT-BiLSTM-CRF algorithm is employed for model development. Meanwhile, Input data refers to the data used to create a relational model for testing purposes. It consists of training and testing data, which are inputted into the system to produce output in words labeled by the system based on entity class predictions.

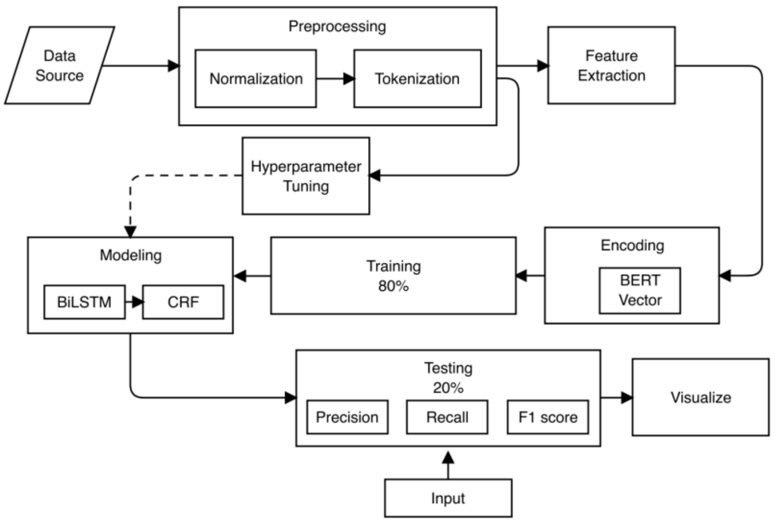


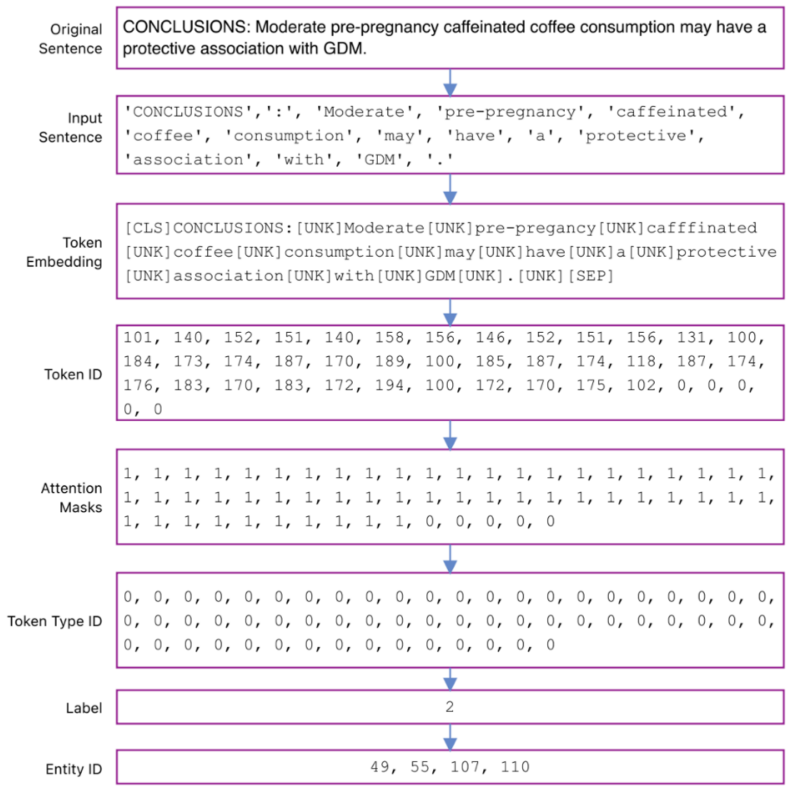
Figure 1. Research stages of this study

2.2. Preprocessing

The preprocessing was conducted by preparing and processing data to make it more manageable. It includes three processes - normalization and tokenization – resulting in labeled words (tokens) that can be used during the feature extraction stage in the dataset training process. The Normalization stage is an initial step in the preprocessing set, where the data normalization is applied to all source data before tokenization. Its goal is to confirm that the tokenised text adheres to the NER and RE format. The text is formatted as one sentence written in a single line and terminated by a period (.), enabling tokenization.

Additionally, this stage involves eliminating meaningless characters like tags (<e1start> and <e1end>). The data normalization process uses the Python library's regular expression (Regex) tools. Regex is a formula for searching the pattern of a sentence or string [12]. Regexes are very helpful in finding sentence patterns; some of the patterns used in regex, like “died?” become "die" or "died" by removing the "?" sign.

Tokenization was performed after the data normalization process. Tokenization is a crucial stage in RE that involves breaking sentences into word pieces, or tokens, for each line [13]. BERT Tokenizer was used to break a sentence into words (tokens). During this stage, the separator between each token is whitespace between characters in sentences, and punctuation characters also serve as separators. The tokenising process in this study uses BERT Tokenizer, which breaks a sentence into words (tokens). Figure 2 shows the tokening process from source data normalization results and input data. BERT represents a word to tackle the phrase that became ambiguous and failed in entity recognition. The input data tokenizing result with BERT Tokenizer is divided into five steps. First, token embedding is breaking a sentence into a token and adding a label [CLS] to mark a token in the first sentence, [SEP] is a token in the end sentence, [UNK] and [PAD] fills between the token. Second, BERT will give a token ID from ‘vocab.json' containing 400,000 vocabulary words; 'vocab.json' will match each token to detect the token position.



CONCLUSIONS: Moderate pre-pregnancy caffeinated coffee consumption may have a protective association with GDM.

```
'CONCLUSIONS',':', 'Moderate', 'pre-pregnancy', 'caffeinated',
'coffee', 'consumption', 'may', 'have', 'a', 'protective',
'association', 'with', 'GDM', '.'
```

```
[CLS]CONCLUSIONS:[UNK]Moderate[UNK]pre-pregancy[UNK]cafffinated
[UNK]coffee[UNK]consumption[UNK]may[UNK]have[UNK]a[UNK]protective
[UNK]association[UNK]with[UNK]GDM[UNK].[UNK][SEP]
```

```
101, 140, 152, 151, 140, 158, 156, 146, 152, 151, 156, 131, 100,
184, 173, 174, 187, 170, 189, 100, 185, 187, 174, 118, 187, 174,
176, 183, 170, 183, 172, 194, 100, 172, 170, 175, 102, 0, 0, 0,
0, 0
```

```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```
2
```

```
49, 55, 107, 110
```

Figure 2. Tokenizing process

Third, the attention mask consists of 0 and 1. Number 0 is the token that is not marked, and number 1 is the position marked to identify the tensor. Fourth, token type id should be used when inputting data more than a sentence; tokens scored 0 and 1. 0 for the first sentence and 1 for the second sentence. Fifth, the label dictionary is the dictionary of RE for keeping relations between entities in the sentence. Sixth, entity id is the position between entities used to know entity position, detected in the n-th column by tag entity <*start*> and <*end*>.

2.3. Feature Extraction

The feature extraction stage aims to produce features from several existing features to form a classifier. The feature extraction process changes the original data, starting from text and moving to a vector. Deep learning cannot process data with actual data; it must first be converted from text to vector. Character information from data and extracting information can be known through the feature extraction [14]. One example of the feature extraction process is the application of word embedding. This study uses the BERT for word embedding. The Bert-base-cased model (https://storage.googleapis.com/bert_models/2018_11_23/multi_ cased_L-12_H-768_A-12.zip) used for the pre-trained process, converting text data into a vector which has a vocabulary of 400,000 vocabulary words say. The BERT model has Layer 12, Hidden Layer 768, and Attention Head 12.

There are two ways to make BERT suitable for specific tasks: feature extraction and fine-tuning. In the first approach, the model's weights remain fixed, and its pre-learned representations are employed in another model for the task, similar to traditional feature-based methods. In the second approach, the pre-trained model can be made adjustable again and fine-tuned for a new task [15].

2.4. Training Dataset

Dataset training is used to form a classifier to develop relationships between entities. The dataset training process uses two algorithms, Bidirectional Long Short-Term Memory and Conditional Random Field (BiLSTM-CRF), by running several iterations in the form of the epoch, batch size, learning rate, max length, and dropout to produce a model file. The resulting model is in hdf5 (hierarchical data format 5) file format, created using the Keras and Tensorflow libraries. In this study, the dataset used was 80% for training and 20% for testing [16]. This study conducted ten experiments using parameter values according to GridSearchCV's recommendations.

GridSearchCV is tweaking hyperparameters to determine the best values for a particular model. The value of hyperparameters has a substantial impact on model performance. Because there is no method to predict the best values for hyperparameters, we must preferably attempt all possible values to determine the optimal values [17]. Using the parameters that gave the best cross-validation performance, a new model is automatically fitted to the whole training dataset using [18].

2.5. Hyperparameter Tuning

This study applies GridSearchCV calculation results and recommends that the best parameter values are epoch=40, dropout=0.3, batch_size=64, and num_units=128. This study uses GridSearchCV to set hyperparameters in the algorithm to obtain an optimal model. This study also conducted experiments 11-14 to perform parameter tuning on the number of epochs and bath size. This technique aims to prove statements from several studies that state that model performance is affected by data pre-processing, amount of data, data quality [19], model complexity, regularisation methods, optimization methods, model architecture, hyperparameters [20], learning methods, and computational capabilities [21].

2.6. BERT-BiLSTM-CRF Architecture

The proposed new architecture aims to improve the model's performance in predicting multi-class in the Plant-Disease Relation that [2] as a benchmark and to resolve the limitations of Kim's research, as explained in the previous section (Figure 3). The extraction relation architecture workflow through the BERT-BILSTM-CRF approach can be divided into several stages. First, after the tokenization process, the data is processed into the BERT Encoder layer, which aims to prevent unambiguous words and errors in entity recognition so that the relationship between entities can be detected.

The input for BERT comprises Token Embeddings, Segment Embeddings, and Position Embeddings, followed by pre-training. The LSTM layer comprises four interconnected neural network layers, increasing its complexity and requiring additional computational time, resulting in an improved model output [22]. The CRF model performs sequence annotation at the sentence level, leveraging dependency information between tags to predict their relationships accurately [23].

Here are the steps in the pseudocode to understand the program flow in predicting entities and plant-disease relationships using the BERT-BiLSTM-CRF approach.

Input: word embedding pre-trained X:
Output: Probabilities of label sequence $P(x|y)$ from sentence input, class prediction (a | b)

Variable
D = collection of documents
S = collection of sentences
X= collection data
E = collection of entities
$W2V$ = word to vector
PE = position embedding
W = weights in the specific dimension space, $W \in \mathbb{R}^d$
Z = weights calculation
σ = sigmoid activation
f = forget gate
i = input gate
h_t = hidden state
b = bias
$\tilde{c}_t$ = candidate state
c_t = current state
o = output gate
$tanh$ = tanh activation
L = sequence of the sentences
$\mathbb{Y}$ = probabilities of the label sequence
y = label target
H = hidden words
(1) Step 1: Embedding Layer

```
(2)          $R$ space dimension
(3)          $D$ segmented to $S$
(4)          $S$ segmented to $X$
(5)          $W$ tokenised into $w_i$ and $e_i$
(6)          $w_i$ and $e_i$ converted to vector ($W2V, PE, POS$) in the dimension space $\mathbb{R}^d$
(7)          Calculated each weight W in the space $W^{W2V}, W^{PE}$
(8)          $Z = W^{W2V}, W^{PE}$
(9)     Step 2: BiLSTM Layer (forward + backward)
(10)         $f_t = \sigma(W_f * [Z_t; h_{t-1}] + b_f)$
(11)         $i_t = \sigma(W_i * [Z_t; h_{t-1}] + b_i)$
(12)         $\tilde{c}_t = tanh(W_c[Z_t; h_{t-1}] + b_c)$
(13)         $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$
(14)         $o_t = \sigma(W_o * [Z_t; h_{t-1}] + b_o)$
(15)         $h_t = o_t * tanh(c_t)$
(16)    Step 3: Conditional Random Field layer
             $p(y|H) = \frac{\prod_{i=1}^{L} \psi_1(y_{i-1}, y_i, h_i)}{\sum_{y' \in \mathbb{Y}} \prod_{i=1}^{L} \psi_1(y'_{i-1}, y'_i, h_i)}$
```

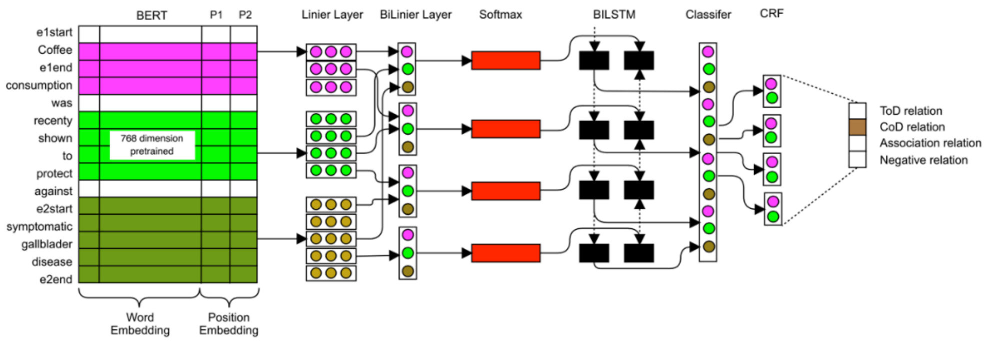


Figure 3. The architecture of BERT-BiLSTM-CRF for plant disease relation task

Each token is written with *Token 1*, *Token 2*, ..., *Token n*, and the *i*-th entity into the text represented by $E_i = E_i^1, \dots, E_i^k$, where E_i^k is the label that has the entity, and $E_i^k = \{Ta \dots Tb\}$ are tagged tokens. Before entering into the encoder process, [CLS] and [SEP] tokens are written into the initial and final sequence represented by $H = \{H_1, \dots, H_2, \dots, H_n\}$, which is formulated $\{H_1, H_2, \dots, H_n\} = BERT(\{T_1, T_2, \dots, T_n\})$. Furthermore, each label with an entity class as the target (predicted) in the input will identify its relationship. For example, a *k* entity in the text predicts the relationship formulated kx (k_1). The number of entities can be two or more to predict the relationship between these entities. This study uses one-pass relation prediction for all pairs of entities simultaneously.

After the BERT Encoder calculates the hidden information from H, the next process is to enter into the linear layer to obtain word embedding H_i from Token T_i (Figure 3). The linear layer is the full connection layer that extracts keyword information from the BERT output. For example, *Hello = Linear(Hello)*. Each entity is represented, and the average and the average embedding token are calculated. The calculation process uses the equation 1.

$$e_i = \frac{1}{j} \sum_{k=1}^{j} e_i^k \tag{1}$$

In equation 1, e_i is the embedding entity of i, e_i^k is the label with entity e_i, and j is the number in entity e_i. Each embedding of the e_i^k label is calculated using equation 2.

$$e_i = \frac{1}{m} \sum_{l=1}^{j} (h_l^{e_i^k}) \tag{2}$$

Refer to equation 2, $h_l^{e_i^k}$ is the embedding token of *l-token*, *m* is the number for the embedding token of *l-token*. The next process is entering into the bilinear layer (Figure 3), namely the full connection layer, which uses two inputs to get a representation of the entity pair. Entity pairs (e_i, e_j) are represented as equation 3.

$$R(e_i, e_j) = BiLinier(e_i, e_j) \quad (3)$$

Calculation results in the bilinear layer are forwarded to the softmax layer to predict the relationship of each pair (Figure 3). The softmax function is used as an activation function in the output layer of a neural network. The softmax function converts an input vector into a probability vector representing each class's relative probabilities. In other words, softmax produces a probability distribution that indicates how likely data is included in each class. The softmax function is defined in equation 4.

$$Softmax(\vec{z})i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}} \quad (4)$$

where $\vec{z}$ is the input vector with z_i elements, and the sum is the sum of all the elements in the $\vec{z}$ vector. The exponential function converts each z_i element to a positive value. Then, the result is normalized by dividing by the sum of all the elements converted to that positive value.

Computation results in the softmax layer are forwarded to the BiLSTM layer. This layer has two directions, forward and backward, which are opposite in processing data. This architecture is suitable for recognizing sentence patterns because each word is processed sequentially. The calculations at each gate are in LSTM units at a time *t* shown in equation 5-10 [24].

$$f_t = \left(W_f.[x_t, h_{t-1}] + b_f\right) \quad (5)$$

$$i_t = (W_i.[x_t, h_{t-1}] + b_i) \quad (6)$$

$$\tilde{C}_t = tanh\left(W_{\tilde{C}_t}.[x_t, h_{t-1}] + b_{\tilde{C}_t}\right) \quad (7)$$

$$C_t = f_t.C_{t-1} + i_t.\tilde{C}_t \quad (8)$$

$$o_t = (W_o.[x_t, h_{t-1}] + b_o) \quad (9)$$

$$h_t = \sigma_t.tanh\ (C_t) \quad (10)$$

where f_t = forget gate, i_t = input gate, o_t = output gate, = sigmoid function, W_x = weight for each gate neuron, x_t = input at the current timestamp, h_{t-1} = output of previous LSTM block (at timestamp *t - 1*, b_x= bias for each gate, $\tilde{C}_t$= candidate for cell state (memory) at timestamp (*t*), c_t = cell state (memory) at timestamp (*t*), and h_t = hidden state.

There are several stages in LSTM to process input viz; the first stage, the forget gate (*f*), will determine whether the information will be deleted or forwarded. Armed with the conditions h_{t-1} and x_t will decide the values 0 and 1 through sigmoid activation using Equation 5. In the second stage, the input gate (*i*) also uses vector values from conditions h_{t-1}, and x_t, which will update to the cell state using Equation 6. At this stage, a new candidate vector is formed through tanh activation using Equation 7. The third stage was updating the old C_{t-1} cell state to C_t through Equation 8. The final stage, the output gate (*o*), uses the vector values of the conditions h_{t-1}, x_t, and activates the sigmoid function to obtain the output values using Equation 9. Next, please put it on the tanh activation and multiply it by the gate output using Equation 10. The prediction results in the forward and backward are combined, then forwarded to the relation classifier to separate classes based on the predictions of the cause of disease (COD), treatment of disease (TOD), association, and negative classes.

The final process of BiLSTM is entered into the CRF layer to improve the prediction process (Figure 3). CRF is a probabilistic model for segmenting and labeling sequential data [25]. CRF combines features to obtain a model to assess sentence predictions with the cause of disease (COD), treatment of disease (TOD), association, and negative relationships. In the CRF layer, the final prediction of BiLSTM, which has a relation class, is segmented and labeled data sequentially by combining the prediction results and providing true label and predicted label information.

2.7. Evaluating benchmarks

The confusion matrix calculates the frequency of every possibility from the predictions made by the classifier [26]. This study uses the micro F1, micro Precision, and micro Recall metrics to test the model's performance [27]. Precision is the value of the prediction accuracy correctly. The precision calculation compares the number of correct predictions with the total number of predictions. Recall is the value of comparing the accuracy of the correct prediction with the total number of predictions that should be correct. Meanwhile, the F1 Score calculates the balance value of Precision and Recall. The Precision, Recall, and F1 scores were calculated as follows:

$$Precision_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fp_i)} \tag{11}$$

$$Recall_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fn_i)} \tag{12}$$

$$F1scores_{\mu} = \frac{(\beta^2 + 1)Precision_{\mu}.Recall_{\mu}}{\beta^2 Precision_{\mu} + Recall_{\mu}} \tag{13}$$

where $\sum_{i=1}^{l} tp_i$is the amount of true positive in whole class, while $\sum_{i=1}^{l}(tp_i + fp_i)$ is the total summation of true positive and false positive. The positive coefficient is employed to balance the mistake types fn and fp. If no inclination is recognized or prearranged, this factor is typically established as 1.

3. RESULTS AND DISCUSSION

In this section, we described the result of preprocessing experiment settings and systematically evaluated our approach's performance on the corpus.

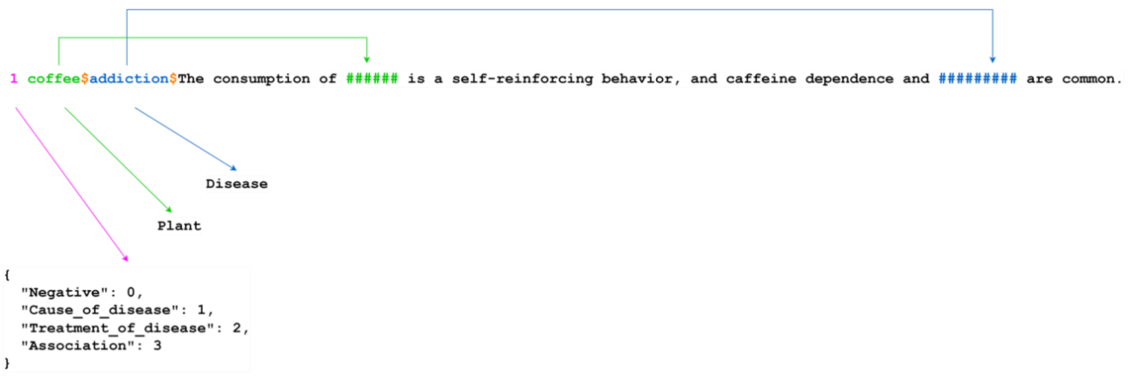
3.1. Data Normalization Result

The normalization process is performed on all source data before the tokenization process. At this stage, it aims to ensure that all the text is in the format for the RE assignment. The format used is one sentence written in one line and ended by a period (.), so tokenization can be done. At this stage, it also removes less meaningful characters such as tags (*<e1start>* entity *<e1end>*). Table 2 shows the results of the data normalization process; the first column is the original sentence from Kim's previous research. The second column removes the *<e1start>* entity *<e1end>* tag and gives the label id at the beginning of the sentence, and then, at the end of the sentence, adds the id for each entity (plant and disease). The entity id is the position between entities that is used to find out the position of the detected entity in the nth column by marking the start and end of the entity. Entities marked with the code *<e1start>* entity *<e1end>* and *<e2start>* entity *<e2end>* are translated to character position in 59 77 81 89.

Table 2. Result of Data Normalization

Orginal	Normalization step 1	Normalization step 2
However, more studies need to further explore the roles of <e1start> vitex agnus castus <e1end> in <e2start> fracture <e2end> repair processes.	0 However, more studies need to further explore the roles of vitex agnus castus in fracture repair processes. 59 77 81 89	0 vitex agnus castus$fracture$ However, more studies need to further explore the roles of ################## in ######## repair processes

Then, in the 3rd column, two marked entities are written at the beginning of the sentence and separated by a string sign ($), then filled with words and replaced the entities in the sentence with a slash (#) as many as the number of entity characters. Figure 4 shows an illustration of the BERT tokenizer-compatible data formats.



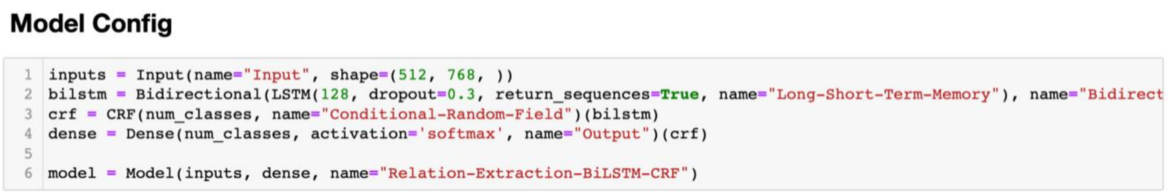
```
1 coffee$addiction$The consumption of ###### is a self-reinforcing behavior, and caffeine dependence and ######### are common.
```

```
{
  "Negative": 0,
  "Cause_of_disease": 1,
  "Treatment_of_disease": 2,
  "Association": 3
}
```

Figure 4. Illustration of BERT data format

3.2. BERT-BiLSTM-CRF Model

This study uses the BERT model, which has 768 hidden layers. Therefore, the model configuration needs to be adjusted. Figure 3 shows the configuration of the input, BiLSTM, and CRF layers. The results of setting the parameters for each layer produce training data in vector form, as many as 928,292 parameters, as shown in Table 3. This study turns off the `early_stopping` parameter to avoid building a model with repeated accuracy scores. As an illustration, at `epoch=20`, it gets an accuracy score of 0.776, and then at `epoch=25`, it receives the same score. Automatically, the algorithm will stop. Even though there are still 15 more iterations, the accuracy score may increase from before.



Model Config

```
inputs = Input(name="Input", shape=(512, 768, ))
bilstm = Bidirectional(LSTM(128, dropout=0.3, return_sequences=True, name="Long-Short-Term-Memory"), name="Bidirect
crf = CRF(num_classes, name="Conditional-Random-Field")(bilstm)
dense = Dense(num_classes, activation='softmax', name="Output")(crf)

model = Model(inputs, dense, name="Relation-Extraction-BiLSTM-CRF")
```

Figure 5. Source code snippet for model configuration

The complexity of the proposed algorithm is by the complexity analysis from the aspect of storage space. This study enables the parameter `save_best_only=True` to save only high-performance models. This approach aims to reduce storage space so that not all epochs that generate models are saved. Some parameter settings are shown in Table 5. Figure 6 shows that data training and validation accuracy have a similar pattern; the model's performance improves as the number of epochs increases. The figure also shows that the model performance is stable at `epoch=30`. Therefore, this study tries to modify the parameters based on these values.

Table 3. BERT-BiLSTM-CRF model summary

Layer (type)	Output shape	Param
Input (input layer)	(None, 512, 768)	0
BiLSTM	(None, 512, 256)	918528
CRF	(None, 768)	8736
Output (Dense)	(None, 4)	1028
Total params: 928,292		
Trainable params: 928,292		
Non-trainable params: 0		

3.2. Performance Calculation

Table 4 shows the experimental results of 10 attempts using the same parameter values. This study only presents a minimum accuracy score of 0.77. Based on the table, it was found that the seventh attempt had the best performance compared to the other experiments. This indicates that this experiment exceeded previous research conducted by [2]. The macro F1 and weighted average metrics determine model performance on unbalanced data [28]. Based on these scores, the "Treatment of Disease" class is interesting to analyse because it is always superior to the others on all metrics used, even though the amount of data is less than the "Negative" class.

Table 4. Result of model performance

Attempt	Class	Precision	Recall	F1 score	Support	Time
6	Negative	0.76	0.73	0.74	118	27.4 minutes
	Cause of Disease	0.65	0.65	0.65	37	
	Treatment of Disease	0.81	0.87	0.84	98	
	Association	1.00	0.75	0.86	8	
	accuracy			**0.77**	261	
	macro avg	0.80	0.75	0.77	261	
	weighted avg	0.77	0.77	0.77	261	
7	Negative	0.77	0.78	0.77	118	26.7 minutes
	Cause of Disease	0.66	0.57	0.61	37	
	Treatment of Disease	0.83	0.89	0.86	98	
	Association	0.75	0.38	0.50	8	
	accuracy			**0.78**	261	
	macro avg	0.75	0.65	0.68	261	
	weighted avg	0.77	0.78	0.77	261	

We assume that the training data influence the achievement of this score in the Treatment of Disease class, which is better than the other classes, even though the number is less than the Negative class. This score indicates that the model's performance is affected by the amount of data, model complexity, learning methods, architectural models, regularisation methods, optimization methods, hardware, and implementation [29]. In addition to the achievements that have been obtained, this study also wants to compare experimental results by modifying batch size and epoch parameters. This study conducted 11-14 experiments with different parameter values but only presented a minimum accuracy score of 0.79, as shown in Table 5.

Table 5. Result of model performance using hyper-parameter tuning

Attempt	Class	Precision	Recall	F1 score	Support	Time
12 epoch=30, batch_size=64, dropout=0.3	Negative	0.81	0.74	0.77	118	21.9 minutes
	Cause of Disease	0.67	0.70	0.68	37	
	Treatment of Disease	0.82	0.91	0.86	98	
	Association	1.00	0.62	0.77	8	
	accuracy			**0.79**	261	
	macro avg	0.82	0.74	0.77	261	
	weighted avg	0.80	0.79	0.79	261	
14 epoch=40, batch_size=16, dropout=0.3	Negative	0.80	0.76	0.78	118	54 minutes
	Cause of Disease	0.66	0.68	0.67	37	
	Treatment of Disease	0.84	0.92	0.88	98	
	Association	1.00	0.38	0.55	8	
	accuracy			**0.80**	261	
	macro avg	0.82	0.68	0.72	261	
	weighted avg	0.80	0.80	0.79	261	

The hyperparameter tuning proposed in this study improved the model's performance by achieving an F-1 micro score of 0.79 in experiment 12. The value of the parameter recommended by GridSearchCV only sometimes gives the best results, but it can be used as a guideline for building models through deep learning. Table 6 compares the model in previous research with the model proposed in this study. The BERT Embedding used in this study proved quite effective and produced a better model than state-of-the-art (SOTA).

Table 6. Comparison of the experimental results of previous research and our model

Data	Model	Embedding	Micro F1
Relation with/without a trigger (1,309 relations)	SDP-CNN	position indicator + position embedding + POS	0.764
Relation with/without a trigger (1,309 relations)	BERT-BiLSTM-CRF (our model)	position indicator + position embedding	**0,790**

BiLSTM, with its ability to capture information from past and future contexts, tends to better capture long-term dependencies in text. This is useful when important dependencies or patterns involving words are located far from each other in the sentence. SDP may have limitations in capturing long-term dependencies, especially if the selected dependency paths cannot cover the required context.

4. CONCLUSION

The problem of predicting PDR was overcome by the BERT-BILSTM-CRF method. The dataset used in the data training influenced the performance calculation model. Even though it used the same parameters, because the training data was taken randomly, it produced different performances in each calculation. In addition, the hyperparameters we used also played an essential role in enhancing the model. This model was due to the other parameters in the BILSTM and CRF layers. The performance evaluation results of the BERT-BiLSTM-CRF model obtained an F1 micro score of 0.790, which proved to be higher than the SDP-CNN model proposed by [2], with a score of 0.764, showing an improvement of 0.260. However, this research still requires further study in the areas 1) ensuring the model can accurately predict multi-classes by attempting BioBERT [30] and BioMedBERT embedding [31], 2) applying transfer learning to detect the plant in scientific name. This research has limitations related to the amount of data used for the modeling process and the unbalanced class distribution. This results in overfitting in the majority class.

ACKNOWLEDGMENTS

Thank you to the National Research and Innovation Agency, Republic of Indonesia, for providing the Degree by Research Scholarship.

REFERENCES

[1] S. Wadhwa, S. Amir, and B. C. Wallace, “Revisiting Relation Extraction in the era of Large Language Models,” *Proc. Annu. Meet. Assoc. Comput. Linguist.*, vol. 1, pp. 15566–15589, 2023, doi: 10.18653/v1/2023.acl-long.868.

[2] B. Kim, W. Choi, and H. Lee, “A corpus of plant–disease relations in the biomedical domain,” *PLoS One*, vol. 14, no. 8, pp. 1–19, 2019, doi: 10.1371/journal.pone.0221582.

[3] Y. Liu, Y. Hou, W. Xu, M. Luo, and X. Sun, “Text Analysis of Community Governance Case based on Entity and Relation Extraction,” *Proc. - 2020 Chinese Autom. Congr. CAC 2020*, pp. 7079–7083, 2020, doi: 10.1109/CAC51589.2020.9327296.

[4] N. Perera, M. Dehmer, F. Emmert-streib, and F. Emmert-streib, “Named Entity Recognition and Relation Detection for Biomedical Information Extraction,” *Front. Cell Dev. Biol.*, vol. 8, no. August, 2020, doi: 10.3389/fcell.2020.00673.

[5] K. Shuang, Y. Tan, Z. Cai, and Y. Sun, “Natural language modeling with syntactic structure dependency,” *Inf. Sci. (Ny).*, vol. 523, pp. 220–233, 2020, doi: 10.1016/j.ins.2020.03.022.

[6] J. Li, K. Shuang, J. Guo, Z. Shi, and H. Wang, “Enhancing Semantic Relation Classification With Shortest Dependency Path Reasoning,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 31, pp. 1550–1560, 2023, doi: 10.1109/TASLP.2023.3265205.

[7] A. P. Ben Veyseh, F. Dernoncourt, D. Dou, and T. H. Nguyen, “Exploiting the syntax-model consistency for neural relation extraction,” *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 8021–8032, 2020, doi: 10.18653/v1/2020.acl-main.715.

[8] H. Yuan, J. Hu, Y. Song, Y. Li, and J. Du, “A new exact algorithm for the shortest path problem: An optimized shortest distance matrix,” *Comput. Ind. Eng.*, vol. 158, no. March, p. 107407, 2021, doi: 10.1016/j.cie.2021.107407.

[9] A. Dash, A. Mohanty, and S. Ghosh, “Advanced NLP Based Entity Key Phrase Extraction and Text-Based Similarity Measures in Hadoop Environment,” *2023 6th Int. Conf. Inf. Syst. Comput. Networks, ISCON 2023*, pp. 1–6, 2023, doi: 10.1109/ISCON57294.2023.10112121.

[10] Q. Li *et al.*, “A Survey on Deep Learning Event Extraction: Approaches and Applications,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 14, no. 9, pp. 1–22, 2022, doi: 10.1109/TNNLS.2022.3213168.

[11] K. Adnan and R. Akbar, “Limitations of information extraction methods and techniques for heterogeneous unstructured big data,” *Int. J. Eng. Bus. Manag.*, vol. 11, pp. 1–23, 2019, doi: 10.1177/1847979019890771.

[12] N. Chida and T. Terauchi, “Repairing Regular Expressions for Extraction,” *Proc. ACM Program. Lang.*, vol. 7, no. June, 2023, doi: 10.1145/3591287.

[13] J. Petrus, Ermatita, Sukemi, and Erwin, “A Novel Approach: Tokenization Framework based on Sentence Structure in Indonesian Language,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 2, pp. 541–549, 2023, doi: 10.14569/IJACSA.2023.0140264.

[14] L. Xue, H. Cao, F. Ye, and Y. Qin, “A method of chinese tourism named entity recognition based on bblc model,” *Proc. - 2019 IEEE SmartWorld, Ubiquitous Intell. Comput. Adv. Trust. Comput. Scalable Comput. Commun. Internet People Smart City Innov. SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, no. September 1995, pp. 1722–1727, 2019, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00307.

[15] M. E. Peters, S. Ruder, and N. A. Smith, “To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks,” in *Proceedings of the 4th Workshop on Representation Learning for NLP*, 2019, pp. 7–14.

[16] V. R. Joseph, “Optimal ratio for data splitting,” *Stat. Anal. Data Min.*, vol. 15, no. 4, pp. 531–538, 2022, doi: 10.1002/sam.11583.

[17] G. L. Team, “An Introduction to GridSearchCV | What is Grid Search | Great Learning,” 2023. https://www.mygreatlearning.com/blog/gridsearchcv/ (accessed Sep. 20, 2023).

[18] Z. M. Alhakeem, Y. M. Jebur, S. N. Henedy, H. Imran, L. F. A. Bernardo, and H. M. Hussein, “Prediction of Ecofriendly Concrete Compressive Strength Using Gradient Boosting Regression Tree Combined with GridSearchCV Hyperparameter-Optimization Techniques,” *Materials (Basel).*, vol. 15, no. 21, 2022, doi: 10.3390/ma15217432.

[19] J. Brownlee, *Machine Learning Mastery with Python: Understand You Data, Create Accurate Models and Work Projects End-to-End*. 2021.

[20] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.

[21] D. Mateja and A. Heinzl, “Towards Machine Learning as an Enabler of Computational Creativity,” *IEEE Trans. Artif. Intell.*, vol. 2, no. 6, pp. 460–475, 2021, doi: 10.1109/TAI.2021.3100456.

[22] H. Tu, L. Han, and G. Nenadic, “Extraction of Medication and Temporal Relation from Clinical Text using Neural Language Models,” *Proc. - 2023 IEEE Int. Conf. Big Data, BigData 2023*, pp. 2735–2744, 2023, doi: 10.1109/BigData59044.2023.10386489.

[23] H. K. Wang, Y. Zhang, and M. Huang, “A conditional random field based feature learning framework for battery capacity prediction,” *Sci. Rep.*, vol. 12, no. 1, pp. 1–12, 2022, doi: 10.1038/s41598-022-17455-x.

[24] Z. Q. Geng, G. F. Chen, Y. M. Han, G. Lu, and F. Li, "Semantic relation extraction using sequential and tree-structured LSTM with attention," *Inf. Sci. (Ny).*, vol. 509, pp. 183–192, 2020, doi: 10.1016/j.ins.2019.09.006.

[25] J. Lafferty, A. Mccallum, and F. Pereira, "Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data Abstract," vol. 2001, no. June, pp. 282–289, 1999.

[26] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics*. The MIT Press, 2015.

[27] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009, doi: 10.1016/j.ipm.2009.03.002.

[28] J. A. Kumar and S. Abirami, "Ensemble application of bidirectional LSTM and GRU for aspect category detection with imbalanced data," *Neural Comput. Appl.*, vol. 33, no. 21, pp. 14603–14621, 2021, doi: 10.1007/s00521-021-06100-9.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2016.

[30] J. Lee *et al.*, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020, doi: 10.1093/bioinformatics/btz682.

[31] S. Chakraborty, E. Bisong, S. Bhatt, T. O. Wagner, F. Mosconi, and R. D. Elliott, "BioMedBERT: A Pre-trained Biomedical Language Model for QA and IR," *COLING 2020 - 28th Int. Conf. Comput. Linguist. Proc. Conf.*, pp. 669–679, 2020, doi: 10.18653/v1/2020.coling-main.59.

BIOGRAPHY OF AUTHORS

Slamet Riyanto is a student in the Postgraduate Program at the Department of Computer Science, Faculty of Mathematics and Natural Sciences, IPB University, Indonesia. Apart from being a student, he has worked as a researcher at the Research Center for Data Science and Information, National Research and Innovation Agency since 2011. His areas of specialization are Text Mining, Natural Language Processing, Information Extraction, and Information Systems.

Imas Sukaesih Sitanggang. She received her Ph.D. in Computer Science from the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia in 2013. She is a lecturer in the Computer Science Department at IPB University, Indonesia. Her main research interests include spatial data mining. She can be contacted at email:imas.sitanggang@apps.ipb.ac.id.

Taufik Djatna is a lecturer and Professor at the Department of Agro-Industrial Technology, Faculty of Agricultural Technology, IPB University, Indonesia. His research interests include blockchain engineering, Kansei engineering, recommendation systems, cyber-physical systems, and customer relationship management. His educational background includes a Bachelor's degree from IPB University, Indonesia, a Master's degree from IPB University, Indonesia, Doctoral degree from Hiroshima University, Japan. He can be contacted at email:taufikdjatna@apps.ipb.ac.id.

Tika Dewi Atikah. She graduated with a doctorate from Graduate School of Environmental Science, Hokkaido University, Japan. She is a researcher at the Research Center Ecology and Ethnobiology, BRIN. Her career in plant research is about the conservation aspect of plants in the community, plant architecture, and demography as well. She can contact at email:tika002@brin.go.id.