Intelligent Chatbot Adapted from Question and Answer System Using RNN-LSTM Model

Intelligent Chatbot Adapted from Question and Answer System Using RNN-LSTM Model

P Anki[1*], A Bustamam[1], H S Al-Ash[1] and D Sarwinda[1]

[1] Department of Mathematics, Universitas Indonesia Depok 16424, Indonesia

Email: prasnurzaki.anki@sci.ui.ac.id*

Abstract. In modern times, the chatbot is implemented to store data collected through a question and answer system, which can be applied in the Python program. The data to be used in this program is the Cornell Movie Dialog Corpus which is a dataset containing a corpus which contains a large collection of metadata-rich fictional conversations extracted from film scripts. The application of chatbot in the Python program can use various models, the one specifically used in this program is the LSTM. The output results from the chatbot program with the application of the LSTM model are in the form of accuracy, as well as a data set that matches the information that the user enters in the chatbot dialog box input. The choice of models that can be applied is based on data that can affect program performance, with the aim of the program which can determine the high or low level of accuracy that will be generated from the results obtained through a program, which can be a major factor in determining the selected model. Based on the application of the LSTM model into the chatbot, it can be concluded that with all program test results consisting of a variety of different parameter pairs, it is stated that Parameter Pair 1 (size_layer 512, num_layers 2, embedded_size 256, learning_rate 0.001, batch_size 32, epoch 20) from File 3 is the LSTM Chatbot with the avg accuracy value of 0.994869 which uses the LSTM model is the best parameter pair.

1. Introduction

Issues submitted by consumers, in general, can be accessed through a number of questions that have been through a question and answer system, which can relate to various storage in data, limited customer service that cannot function fully for a whole day, so the program is needed to work optimally and produce service results in the form of answers to questions raised by consumers. Choosing chatbots as a solution for answers of questions based on various problems that consumers issue can make it easier for consumers to get answers. In arranging the structure of the display components on the chatbot into the Python program, because it is easy to use, and more productive in interpretation programs [1]. Based on set of sequences, computational methods must solve two major problems: effectively representing a sequence as a feature vector that can be analyzed and designing a model that can identify data accurately and quickly [2]. In the end, it is hoped that the input data generated through the question and answer system will be prepared, which will then be implemented into the Python program so that in the end, the consumers can get answers to the questions they raised to the chatbot. Chatbot is a system that accepts user input with an ongoing response, parts of the chatbot can be built from an encoder-decoder architecture [3]. Simply put, a chatbot is a simple robot in the form of a program to answer questions from users, which produces output data in the form of answers. In spite of users' low satisfaction and continuance intention (CI) regarding chatbots, few studies have explored why consumers are reluctant to continue using them. In light of this, empirical investigation of the users' satisfaction with chatbots and its CI becomes relevant [4]. When the research results can understand the information needs needed

by consumers with the linkage of information systems in the chatbot, it is expected that consumers' satisfaction will also increase.

There is a high degree of variance in chatbot quality. Our studies compare a hypothetically perfect chatbot (error-free) to a chatbot which struggles to infer meaning and thus seeks clarification (clarification) to a third chatbot which produces an error in comprehension (error). Predicting that the error-free chatbot will outperform the error producing chatbot is straightforward [5]. In this research will compare how accurately the chatbot can meet the information needs required by the user, so it can also be seen how much error is caused due to the mismatch of the information required. In section 2, we will discuss about materials and method will be used in this research, among others are steps in making chatbot, which will tell the reader what steps are required to create a chatbot, discussion regarding LSTM models, which will provide info regarding what the LSTM model is and how it is implemented, displays a study of the greedy method, which will help improve program performance, conduct lessons related to the seq2seq model, to help translate sentences in the program.

2. Materials and method

There are set of sentences from dataset, which can used to build chatbot program based on LSTM model, multiple parameter pairs, Greedy method. Input from the user can be command to run chatbot program, with the result that set of sentences which contain information according to the input that the user enters.

2.1. Steps in making a chatbot

There are 4 steps to creating a chatbot, namely: data identity, data input for the question and answer system, compiling a chatbot program, and evaluating the output.

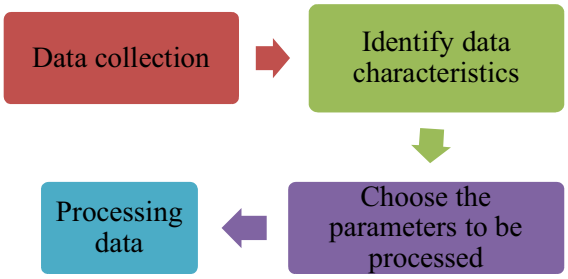

Figure 1. Data management diagram

- Data identity

The data to be used in this program is the Cornell Movie Dialog Corpus which is a dataset containing a corpus which contains a large collection of metadata-rich fictional conversations extracted from film scripts, which have 220,579 conversation exchanges between 10,292 pairs of movie characters, involving 9,035 characters from 617 films, and a total of 304,713 sayings [6]. The data used is the 2018 data. Based on the parameter selection stage, the conversation data will then be selected and then processed from the input questions that come from the dataset, which will then be answered by the machine as a form of response to the chatbot program. The answers to the questions will produce a response for individuals who want to obtain answers about matters related to the characteristics of film data. To manage this data, below is a data management diagram that explains the steps from collecting to processing the data.

- Question and answer system input data

In carrying out input from program users, files containing dialog sentences in the film are inputted into the program as user input. Then, the input will be processed to obtain the output of the program in the form of a dialogue sentence in the film, by having a relationship between the dialog in the input to the dialog in the output.

- Chatbot program development

The following are the things that need to be considered in the preparation of the chatbot program: there will be various translation choices in the form of sequence-to-sequence, and the selection of the LSTM model so that we can determine the best accuracy from the response of the chatbot compared to the live human response.

- Output evaluation

The last step is evaluating whether or not the model has provided accurate results. For example, there may be a relationship between the input dialog and the output dialog that turns out to be much more accurate than an input dialog selection that the program user previously entered.

2.2. LSTM Model

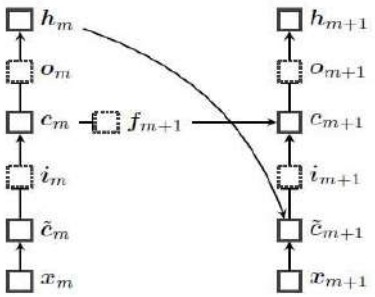

Figure 2. LSTM architecture [3]

The Long Short-Term Memory (LSTM) model is a special type of RNN that has the ability to study long-term dependencies [7]. In this model, the encoder is applied to the last hidden statement of the LSTM [3]. The LSTM architecture is shown in Figure 2, and the full equation for updates is as follows from [3]:

$$f_{m+1} = \sigma\left(\theta^{(h\to f)}h_m + \theta^{(x\to f)}x_{m+1} + b_f\right) \textit{ forget gate} \quad (1)$$

$$i_{m+1} = \sigma(\theta^{(h\to i)}h_m + \theta^{(x\to i)}x_{m+1} + b_i) \textit{ Input gate} \quad (2)$$

$$\tilde{c}_{m+1} = \tanh(\theta^{(h\to c)}h_m + \theta^{(w\to c)}x_{m+1}) \text{ update candidate} \quad (3)$$

$$c_{m+1} = f_{m+1}\odot c_m + i_{m+1}\odot\tilde{c}_{m+1} \text{ memory cell update} \quad (4)$$

$$o_{m+1} = \sigma(\theta^{(h\to o)}h_m + \theta^{(x\to o)}x_{m+1} + b_o \textit{ Output} \text{ gate} \quad (5)$$

$$h_{m+1} = o_{m+1}\odot\tanh(c_{m+1}) \textit{ Output} \quad (6)$$

The operator Θ is an elementwise product. The LSTM model is the result of adding a hidden state which is expressed in $h_m$with the memory cells represented as c_m. The value of the memory cell at each mth time is the gate form which is a sum of two quantities: the value of the previous memory cell expressed with c_{m-1}, and the value of the memory cell that has undergone an update expressed with c_m, which is calculated from the previous input in the current $x_m$form and the previous hidden state in the h_{m-1}, form. The next state is h_m calculated from the cell memory. Since memory cells do not pass through non-linear function pathways during renewal, it is possible for information over remote networks [3].

Each gate is controlled by a vector that has a weight, which determines the previous hidden state (e.g., $\theta^{(h\to f)}$) and Input current (e.g., $\theta^{(x\to f)}$), plus the vector offset (e.g. , b_f). The overall operation can be informally summarized as (h_m, c_m)= LSTM $(x_m, (h_{m-1}, c_{m-1}))$, with (h_m, c_m) representing the LSTM status after reading the m token. LSTM outperforms the standard artificial neural network on

a wide range of issues. The gates are shown in squares with dotted edges. In the LSTM language model, each h_m will be used to predict the next word w_{m+1}. LSTM outperforms standard recurrent neural networks in a variety of problems, for example in language modeling problems [3].

Based on the discussion of [3], Recurrent neural networks (RNNs) were introduced as a language modeling technique, where the context in token m (where the token stops) as well as token M (the specified M token value) is summarized by the repeated updated vector,

$$h_{\boldsymbol{m}} = g(x_m, h_{m-1}), m = 1, 2, \ldots \text{M}, \quad (7)$$

where x_m is the stored vector of the w_m token and the function g defines the loop. The initial condition h_0 is an additional parameter of the model. The LSTM (Long Short Term Memory) model has a more complex loop, in which the memory cells pass through a series of gates, avoiding repeated applications of non-linearity. A Gate is an input function and is the previous hidden status. The form is calculated from the activation of the elementwise sigmoid, $\sigma(\text{x}) = (1 + \exp(-\text{x}))^{-1}$, ensuring that the value will be in the range of [0, 1]. Therefore, this form can be seen as a differentiated logic gate.

The LSTM model is one of many parallel computing models. Based on reference from [8], parallel computing can be used to process the program instructions by distributing them into multiple processors in order to reduce the running time of the program.

2.3. Greedy method

In addition to choosing the model, the next step is to determine the method used in the program. In this program, the greedy method is chosen as a form of implementing the LSTM model so that when running the program, the data processing time can be faster while also increasing the accuracy of the selected model [9]. Based on the discussion of the greedy algorithm in [10], the greedy algorithm basically chooses the best decisions by looking at the closest estimate of each stage of the problem in the hopes of finding a good solution. In this case, the algorithm selects the lowest-cost worker who is then assigned to the task as the first task, then selects the next lowest-cost worker to be assigned to that task, and so on until all the tasks have been assigned. The following is a case example of applying the greedy algorithm for the Linear Assigment Problem (LAP):

- Step 1: Find the location of the smallest element, and delete the location from the rows and columns.
- Step 2: Find the location of the second smallest element, and delete the locations of the rows and columns.
- Step 3: Repeat the process until there are no rows and columns to delete.
- Step 4: Select the given result as the optimal local result.

The greedy algorithm tries to approach the optimal solution by increasing the candidate solution iteratively, without any guarantee that the optimal solution will actually be found. Greedy's algorithm itself will find good answers, but it can scramble parts of the algorithm, but will ensure that many answers are possible. Modifying the greedy algorithm so that it sometimes accepts assignments at certain periods of time which can worsen the objective function, but will then succeed in obtaining local optimizations and possibly finding optimal global solutions.

2.4. Seq2seq model

Things that need to be known regarding the importance of implementing the seq2seq model are as follows:

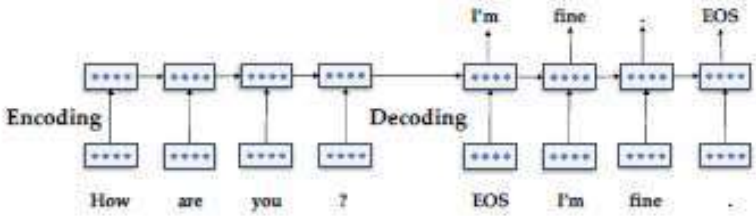

Figure 3. Decoder Encoder Model for Neural Response Generation in Dialogue [9]

For the software used, Jupyter Notebook software based on Python is chosen so that program users can clearly see the input and output of the program being run. When implementing a question and answer system into a program, it is necessary to have a seq2seq model, which can function to produce various responses to user input [9]. Things that need to be known, regarding the importance of implementing the seq2seq model are as follows:

- In its application as the basis of the encoders model, an entered input has a tendency to produce a predictable but repetitive response to close the conversation.
- The problem above can be overcome by changing the objective function for training the seq2seq model into mutual information destinations or by modifying the decoder to make it more diverse in response to the things that are given.

In section 3, we will discuss about data implementation in the python program, which will explain what is needed to create a chatbot program using the python program.

3. Data implementation in the python program

To implement the data into the Python program with the Jupyter Notebook software, a program planning plan will be systematically compiled as follows:

a. Choosing the software to be used

In choosing software, the main things to be considered are things such as whether the selected data will be processed properly in the program, the performance of the software in processing data, and the availability of supporting attributes for programmers needed in making the program.

b. Selecting supporting models and attributes

Choosing a model that is in accordance with the characteristics of the data can affect program performance. Determining a high or low level of accuracy that is generated from a program is a major factor in choosing the model. Based on the consideration of the model selection requirements, the LSTM model was selected as model to be applied into the program. In addition, supporting attributes (such as the seq2seq model), are the next determining factors that can verify whether the data processing process matches the criterias that was meant to guide the process. In application to the program, the seq2seq model can process input sentences which will then be processed with other models and structures in the program, so that in the end it can issue various output sentences as a response generated from the chatbot program.

c. Determining the program evaluation method

In this case, there are various program evaluation methods, such as loss, accuracy, val_loss, and val_accuracy. Based on the case example from [9], to introduce a method for evaluating text classification, some simple binary detection tasks will first be considered.

For example, in spam detection, the aim is to label any text that is in the spam category in the form of a label ("positive") or not in the spam category in the form of another label ("negative"). For each item in the form of an email document, it is necessary to know whether the system created can determine whether an item is spam or not.

What needs to be known to determine whether an email is actually spam or not, for example, the human-defined labels for each document will be tested for suitability. The mention of human labels will then be referred to as the gold table. Before testing spam detection, it is necessary to prepare a standard measurement to find out how well the spam detection has performed. One thing to pay attention to to evaluate any system for detecting something is to start with constructing a Contingency Table as shown in Table 1.

Table 1. Contingency Table [9]

		gold standard labels		
		system positive	*system negative*	*precision* $= \frac{tp}{tp+fp}$
system output labels	*system positive*	*true positive*	*false positive*	
	system negative	*false negative*	*true negative*	
		recall $= \frac{tp}{tp+fn}$		*accuracy* $= \frac{tp+tn}{tp+fp+tn+fn}$

Each cell is labeled as the set of possible outcomes. In the case of spam detection, for example, a true positive is a document that is spam (shown in the Contingency Table) and the result from the system says it is spam. False negatives are documents that are spam but result from the system labeling them non-spam. True negatives are documents that are spam and result from the system labeling them as spam. False positives are documents that are non-spam but the results from the system labels them as spam. In the lower right corner of the Contingency Table there is the form of the equation for accuracy, as follows:

$$accuracy = \frac{tp+t}{tp+fp+tn+f} \quad (8)$$

The following are matters that need to be prepared to obtain the loss value. Based on the theory of [11], for example, for an observation *x*, the loss function is stated as in the equation below:

$$L(\hat{y}, y) = The\ amount\ of\ difference\ from\ the\ true\ y\ value \quad (9)$$

In calculation to determine how close the output classifier ($\hat{y} = \sigma(w.x + b)$) is to the actual output (y, which is 0 or 1).

Based on the definition in [11], validation loss is an estimate of the error rate derived from the model that is calculated based on the loss function of the validation set, which is observed through the training process. After the model has been selected and trained, it is necessary to evaluate how effective this model is for the purposes of the classification task. Intuitively, one way to evaluate a model is to calculate the percentage of samples that have been correctly classified. So, the classification rate can be calculated from the following model:

$$classification\ rate = \frac{number\ of\ samples\ correctly\ classified}{total\ number\ of\ samples} \quad (10)$$

Based on the equation above, classification errors can be calculated and are complementary to the classification rates. The calculation of the error rate of the model can be calculated as follows:

$$error\ rate = \frac{number\ of\ loss\ function\ over\ validations\ set}{total\ number\ of\ validations\ set} \quad (11)$$

The percentage generated by calculating the loss function against the set of validations comes from a predetermined model. Based on the case example from [12], we can see the comparison between the

loss from the training dataset set against the val_loss from the validation set, through the graph in Figure 4.

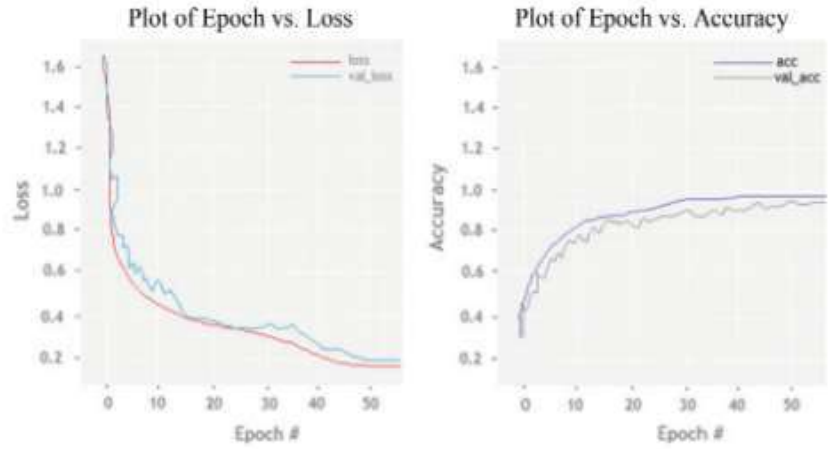

Figure 4. Comparison chart of epoch to loss and comparison chart of epoch to accuracy [12]

Based on the graph in Figure 4, it can be seen that the validation loss (val_loss) can be lower or higher than the loss value of the training dataset. In other words, it can be called underfitted or overfitted. Based on the case example from [12], we can see the comparison between the accuracy of the training dataset (acc) to the validation accuracy (val_acc) of the validation set, through the graph in Figure 4. Because the training dataset is data that is known from the model and validation data is data that is unknown from the model, the acc value is generally higher than val_acc. If the validation accuracy value (val_acc) is lower or higher than the accuracy value of the training dataset, it can be said to be underfitted or overfitted. After deliberating the comparison of the elaboration of each program evaluation method, accuracy is chosen as the appropriate program evaluation method for the chatbot program.

In section 4, we will discuss about applying data implementation in the python program, which will be more detail with description of the problem which will explain how the program description should be created, and program making which will inform the reader how to make chatbot based on flow of programming that has been planned.

4. Applying data implementation in the python program

This elaboration will explain the description of the problem that can be used as a study of the need for implementing chatbots in the question and answer system.

4.1. Description of the problem

In facing the dynamics of modern times, there will indeed be numerous questions that consumers wantto ask, based on things related to a specific data. Based on the speed comparison between manual question and answer service system carried between humans with the resulting response based on the chatbot program, which is the implementation of question and answer data between humans and machines, as well as easy access and operation times according to needs of the chatbot program user, these are the reasons why it is necessary to create a chatbot program, as a form of implementation of the question and answer system data. Before discussing the components needed to make a chatbot, it is necessary to know the things that must be prepared to obtain the components needed for making the program as shown in the diagram below:

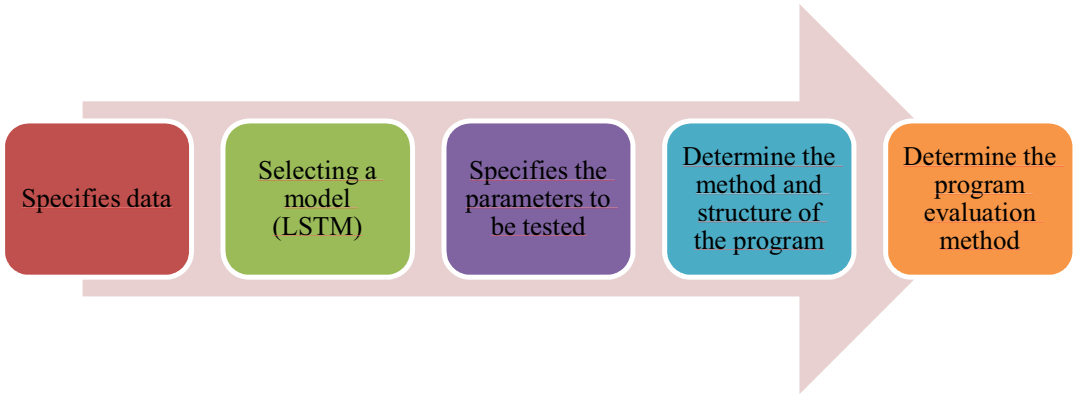

Figure 5. Preparation diagram of the components for the chatbot program

4.2. Program making

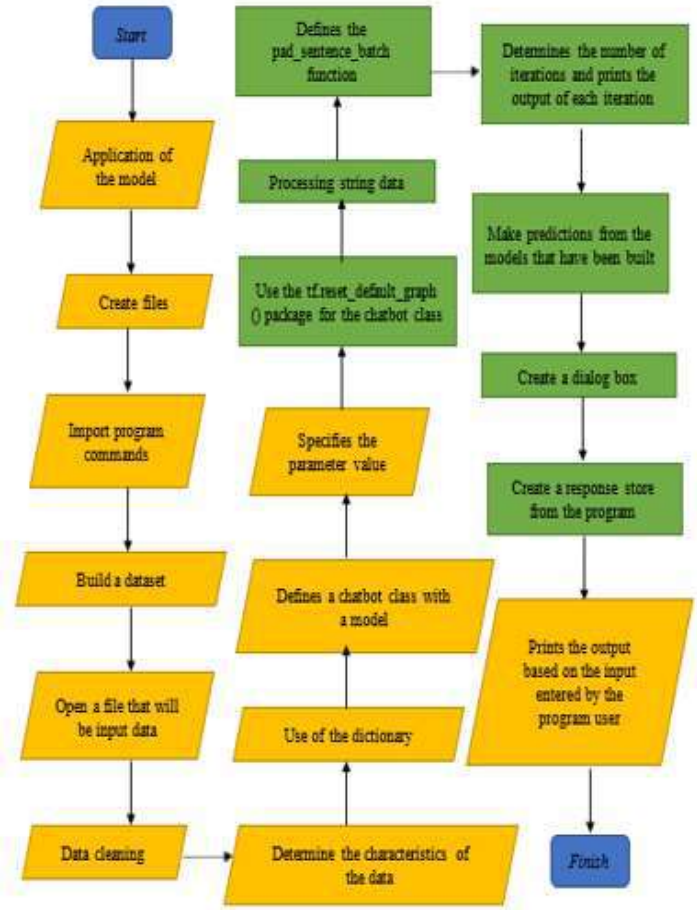

Figure 6. Program making flowchart

In section 5, we will discuss about program creation detail and discussion of test results, in this section will discuss the description of the file that will be named according to the parameter pair used, as well as the test results that have been tested in the program.

5. Program creation detail and discusssion of test results

After the chatbot program has been created, there will be 3 files generated as a result of implementing the LSTM model into the chatbot program, as shown below:

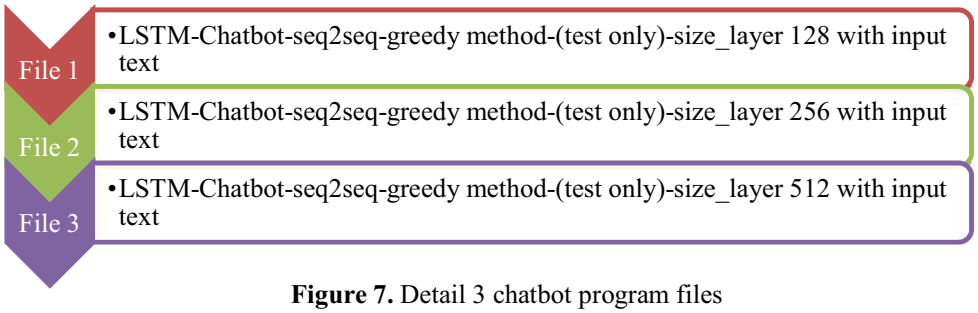

Figure 7. Detail 3 chatbot program files

Table 2. Data to be Tested in LSTM Chatbot

Parameter Pair	File 1	File 2	File 3
size_layer	128	256	512
num_layers	2	2	2
embedded_size	64	128	256
learning_rate	0.001, 0.0015	0.001, 0.0015	0.001, 0.0015
batch_size	8	16	32
epoch	20,30,40,50	20,30,40,50	20,30,40,50

In Table 2, it can see the various parameter pairs that will be tested in the LSTM chatbot program. Furthermore, with a total of 24 different parameter pairs (different numbers are on size_layer, learning_rate, epoch), details of the 8 parameter pairs in 3 different files will be presented with the results of all tested parameter pairs are shown in Figure 8.

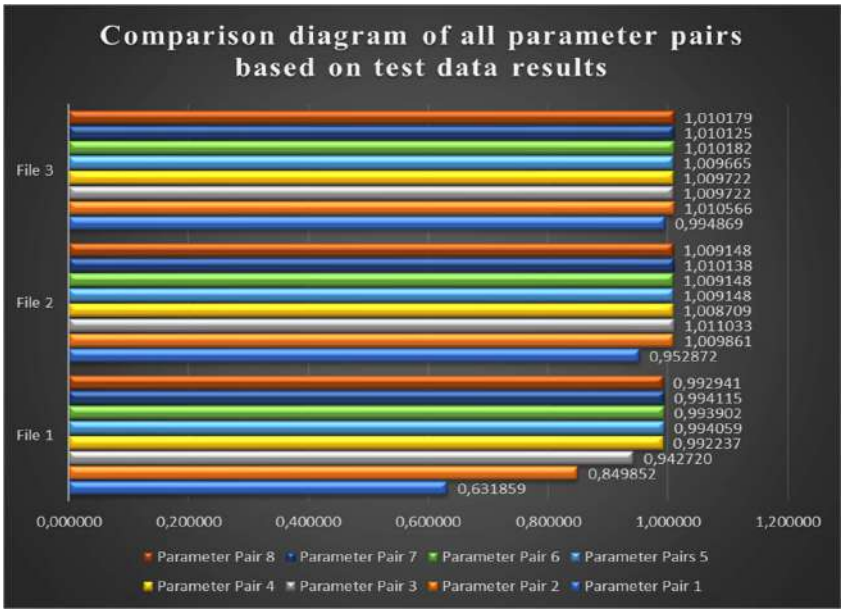

Figure 8. Comparison diagram of all parameter pairs based on test data results

Table 3. Best Test Result Data from Each File

Parameter Type	**File 1** *Parameter Pair 7*	**File 2** *Parameter Pair 1*	**File 3** *Parameter Pair 1*
size_layer	128	256	512
num_layers	2	2	2
embedded_size	64	128	256
learning_rate	0.001	0.001	0.001
batch_size	8	16	32
epoch	50	20	20
avg accuracy	0.994115	0.952872	0.994869

Based on the parameter pairs in each of the following files, the best parameter pair from the 3 files that have been tested will not be assessed if > 1.0 an overfit condition occurs (when the training accuracy results are very good but the testing accuracy results are not as good). What will be selected is the one that produces the best avg accuracy on a scale of 0.0 to 1.0 in the Table 3. So, based on all the test results of the program that has been carried out, it can be stated that the Parameter Pair 1 originating from File 3 is the LSTM Chatbot with the avg accuracy value of 0.994869 is the best parameter pair. The resulting accuracy in the chatbot research using English using the LSTM model is 70.9% [13]. When viewed from these references, in this study it can be stated that there has been an increase in accuracy compared to previous studies.

In section 6, we will discuss about conclusion and future work of this research, which is conclusion of the results in this research, and future work which is relating to things that need to be improved from this research.

6. Conclusion and future work

Based on the application of the LSTM model into the chatbot, it can be concluded that with all program test results consisting of a variety of different parameter pairs, it is stated that Parameter Pair 1 (size_layer 512, num_layers 2, embedded_size 256, learning_rate 0.001, batch_size 32, epoch 20) from File 3 is the LSTM Chatbot with the avg accuracy value of 0.994869 which uses the LSTM model is the best parameter pair. For future works, we can try improving the LSTM chatbot algorithm using advanced computing environment [15].

In the future work, it is expected that there will be an increase in accuracy from the updating of methods, models, and references that are increasingly following technological developments.

7. References

[1] Raj S 2018 *Building Chatbots with Python: Using Natural Language Processing and Machine Learning* (New York City: Apress) p. 33
[2] Bustamam A, Musti M I S, Hartomo S, Aprilia S, Tampubolon P P and Lestari D 2019 Performance of rotation forest ensemble classifier and feature extractor in predicting protein interactions using amino acid sequences *BMC Genomics* vol 20 (London: BMC Genomic) p. 2
[3] Eisenstein J 2018 *Natural Language Processing* (Cambridge: MIT press) pp. 137-138, p. 345
[4] Ashfaq M, Jiang Y, Shubin Y and Loureiro S M C 2020 I, Chatbot: Modeling the determinants of users' satisfaction and continuance intention of AI-powered service agents *Telematics and Informatics* vol 54 (Amsterdam: Elsevier) p. 2
[5] Sheehan B, Hyun S J and Gottlieb U 2020 Customer service chatbots: Anthropomorphism and adoption *Journal of Business Research* vol 115 (Amsterdam: Elsevier) p. 15
[6] Chidananda R 2018 *Cornell Movie-Dialogs Corpus* (Kaggle datasets)
[7] Muzaffar S and Afshari A 2019 Short-Term Load Forecasts Using LSTM Networks *Energy Procedia* vol 158 (Amsterdam: Elsevier) p. 2922

[8] Ardaneswari G, Bustamam A, Siswantining T 2017 Implementation of parallel k-means algorithm for two-phase method biclustering in Carcinoma tumor gene expression data *AIP Conference Proceedings* vol 1825 (Maryland: AIP Conference Proceedings) p. 2
[9] Jurafsky D and Martin J H 2019 *Speech and Language Processing An Introduction to Natural Language Processing* (New Jersey: Prentince Hall) p. 66, p. 81, p. 163, p. 497
[10] Güneri Ö İ, Durmuş B and Aydin D 2019 Different Approaches to Solution of The Assignment Problem Using R Program *Journal of Mathematics and Statistical Science* vol 5 (Delaware: SSPub) p. 134
[11] Clavance L 2019 *An Evaluation of Machine Learning Approaches to Natural Language Processing for Legal Text Classification* (London: Imperial College London)
[12] Kotecha K, Piuri V, Shah H N, Patel R 2020 *Data Science and Intelligent Applications: Proceedings of ICDSIA 2020* (New York: Springer Nature) p. 117
[13] Peters, F 2018 *Master thesis : Design and implementation of a chatbot in the context of customer support* (Belgium: University of Liège) p.47
[14] Muradi H, Bustamam A and Lestari D 2015 Application of hierarchical clustering ordered partitioning and collapsing hybrid in Ebola Virus phylogenetic analysis *2015 International Conference on Advanced Computer Science and Information Systems* (New York City: IEEE) p. 323

Acknowledgments
This research is partially supported by PUTI Prosiding 2020 research grant by DRPM Universitas Indonesia with contract number NKB-927/UN2.RST/HKP.05.00/2020.