TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 18, No. 3, June 2020, pp. 1422~1432
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v18i3.14791

Cleveree: an artificially intelligent web service for Jacob voice chatbot

Octavany, Arya Wicaksana
Department of Informatics, Universitas Multimedia Nusantara, Indonesia

Article Info

Article history:

Received Aug 3, 2019
Revised Jan 20, 2020
Accepted Feb 26, 2020

Keywords:

Cosine similarity
LSTM
Stacked residual
TextRank
Web service

ABSTRACT

Jacob is a voice chatbot that use Wit.ai to get the context of the question and give an answer based on that context. However, Jacob has no variation in answer and could not recognize the context well if it has not been learned previously by the Wit.ai. Thus, this paper proposes two features of artificial intelligence (AI) built as a web service: the paraphrase of answers using the Stacked Residual LSTM model and the question summarization using Cosine Similarity with pre-trained Word2Vec and TextRank algorithm. These two features are novel designs that are tailored to Jacob, this AI module is called Cleveree. The evaluation of Cleveree is carried out using the technology acceptance model (TAM) method and interview with Jacob admins. The results show that 79.17% of respondents strongly agree that both features are useful and 72.57% of respondents strongly agree that both features are easy to use.

Corresponding Author:

Arya Wicaksana,
Department of Informatics,
Universitas Multimedia Nusantara,
Scientia Boulevard St., Gading Serpong, Tangerang-15810, Banten, Indonesia.
Email: arya.wicaksana@umn.ac.id

1. INTRODUCTION

A chatbot is a messaging program that interacts with users like chatting with people. There is research on developing a dialogue system or chatbot using natural language that is useful for education, customer service, and entertainment purposes [1-3]. In Universitas Multimedia Nusantara (UMN), there is a web-based voice chatbot called Jacob which is developed by Wijaya in [4]. Jacob uses the Wit.ai platform to get the context of the user's question by extracting the intent (the goal of the user is coming to the chatbot) and entities (important variable in intent that helps add relevance to an intent) of the question. Therefore, the chatbot could reply to the question based on the context (intent and entities). However, Jacob has two shortcomings when interacting with the user. First, Jacob only replies with answers that are already programmed in its knowledge base, which results in repetitive answers. The second issue is that Jacob sometimes misunderstood the context of the question because it has never been learned before by the Wit.ai platform. This is due to the problem where similar sentences could have the same meaning or context. Thus, the Wit.ai platform could mistakenly give different context or even does not recognize the context at all.

The solution to the first problem is by varying the answers using a paraphrase. In natural language processing (NLP), paraphrases are an interesting task to solve. It is difficult to build a paraphrase recognition system because paraphrases are hard to define [5]. In the linguistic literature, paraphrases are sentences or phrases that have an approximate equivalence of meaning in the different wording [5]. Thus, a deep neural network is used in this study to generate the paraphrase of the answer. For NLP tasks, recurrent neural network (RNN) architecture gives a good performance, especially to a special kind of RNN called Long short-term memory (LSTM), by reducing the perplexity and word error rate [6, 7]. LSTM is widely well known due to its capability of learning long-term dependencies and reducing the vanishing gradient problem. LSTM could be implemented to predict the next word by the previous words in language modeling and generating text and implemented in chatbot application [2]. Yavuz et al. in [8] develop a response generation using LSTM and hierarchical pointer network. Furthermore, there is an LSTM model that is used to generate paraphrase by adding the residue, called stacked residual LSTM [9]. We use this model because it has better results than Sequence to Sequence, Bi-directional LSTM, attention-based LSTM model [9]. The pre-trained model of stacked residual LSTM from [9] is implemented to generate paraphrase of answers in this paper.

In the face of the unexpected results of intent and entities, it is necessary to update the knowledge in the Wit.ai platform from the history of questions. Based on the study about Jacob, there are conversation logs that record all the conversations between Jacob and the user. So, we propose the solution to prevent an administrator to manually read all of the questions in the conversation logs by extracting the summary of questions, which selects the most frequently asked questions from the entire conversation logs [10]. Ideally, the extractive summarization should contain around 20% of the sentences from the entire text [11]. There are six algorithms for extractive summarization that have been evaluated by Victor et al in [12], which are Luhn, TextRank, LexRank, LSA, SumBasic, and KLSum. Based on the results, Luhn and TextRank have the best performance to get the extractive summary for the speech-to-text case [12]. Between those both algorithms, we choose TextRank because the TextRank algorithm is a graph-based ranking algorithm that has been proven to be successful for the identification of the most important or relevant sentences (vertex) in the text (graph) [13]. According to [10, 14] TextRank algorithm with cosine similarity using Word2Vec could enhance the ranking process.

2. RESEARCH METHOD

This section contains a brief explanation of LSTM as the main method implemented in this study. Then, we describe TextRank algorithm, Cosine Similarity, and Word Embedding. In order to that, we also explain our proposed method.

2.1. Long short-term memory (LSTM)

LSTM is a variant of the recurrent network which is different from the feed-forward network and is introduced by [15]. Recurrent network feeds its outputs back into its own inputs so the response of the network to a given input may depend on previous inputs [16]. LTSM computes the hidden state h_t by adding a memory cell C_t at every time step t [9]. When the unit computes the memory cell and hidden state at time step t, it considers the input state at time step t, the hidden state h_{t-1}, and the memory cell C_{t-1} at time step t [9]. LSTM has three gates that each gate consists of one sigmoid layer and pointwise multiplication operation which known as forget gate (f_t), input gate (i_t), and output gate (o_t) [17]. The gates are described in Figure 1.

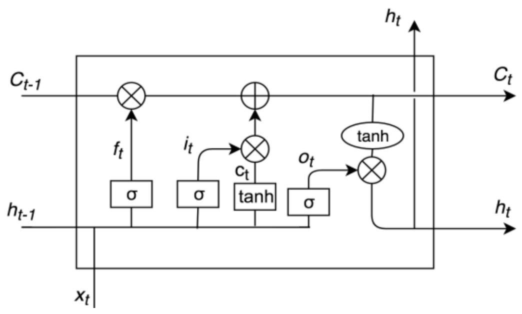


Figure 1. LSTM cell contains forget gate, input gate, new memory cell, and output gate [17]

2.1.1. Stacked residual LSTM

Stacked Residual LSTM is a model that was proposed by Prakash et al. and the addition of residual connections to generate paraphrase and this model can help overcome a degradation problem [9]. Residual connections are added after every n layers as the pointwise addition [9]. Figure 2 shows that the residual connections are added after every two layers as the pointwise addition.

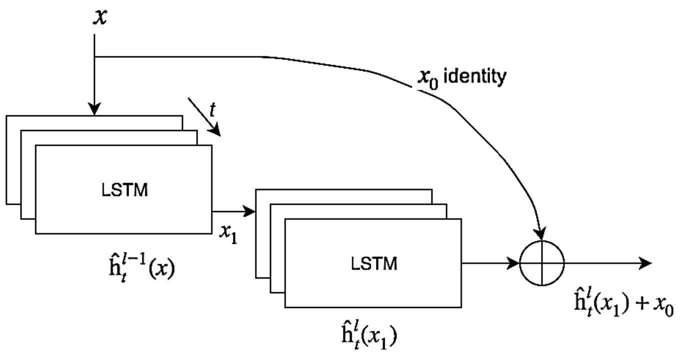


Figure 2. A unit of Stacked Residual LSTM [9]

2.2. TextRank

TextRank is a graph-based ranking algorithm for the identification of the most important or relevant sentences in the text [13]. One vertex represents a sentence and the edges represent the relation score (weight) between two sentences. A vertex with the highest score of TextRank is the most relevant or important in the graph. TextRank is a weighted directed graph $G = (V, E)$, where the graph G consists of a set of vertices V and a set of edges E. For a given vertex V_i, $In(V_i)$ represents the set of vertices that point to vertex V_i, and $Out(V_i)$ represents the set of vertices that point from vertex V_i. The weight from vertex i to vertex j is represented as W_{ij}. The formula of TextRank can be defined as shown in (1) [18],

$$TR(V_i) = (1-d) + d \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} TR(V_j) \quad (1)$$

in (1), $TR(V_i)$ represents the score of vertex V_i; d is the damping factor with value is between 0 to 1. Normally, the damping factor is set to 0.85 [13].

2.2.1. Word embedding

Word embedding is a vector representation of words based on the context of the sentences or semantic relationships between words. The vector contains real number [19]. Mikolov et al. proposed two models that focus on learning word vectors which are continuous bag-of-words (CBoW) and Continuous Skip-gram (SG) as shown in Figure 3. These two models called Word2Vec. CBoW is optimized to predict a word based on words around it or the context. SG is optimized to predict the context based on the current word [20].

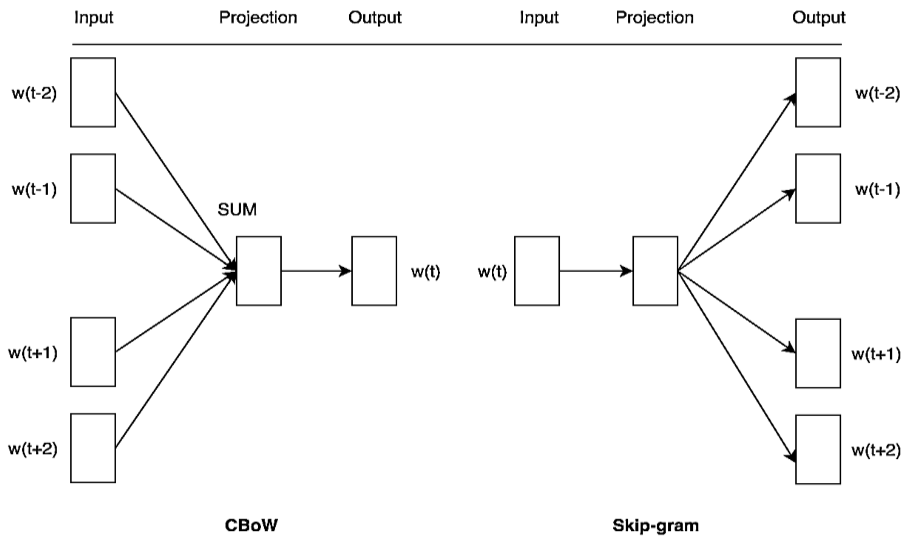


Figure 3. Continuous bag-of-words (left) and skip-gram (right) [20]

2.2.2. Cosine similarity

Cosine Similarity calculates the similarity by measuring the cosine of the angle between two vectors. A result is a number between zero and one. If the result is close to one, the more similar its two vectors [21]. In this paper, the vectors represent the sentence vectors which are calculated using Word2Vec. The similarity between vector $\vec{i}$ and vector $\vec{j}$ can be defined as shown in (2):

$$sim(i,j) = cos\,(\vec{i},\vec{j}) = \frac{\vec{i}.\vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2} \tag{2}$$

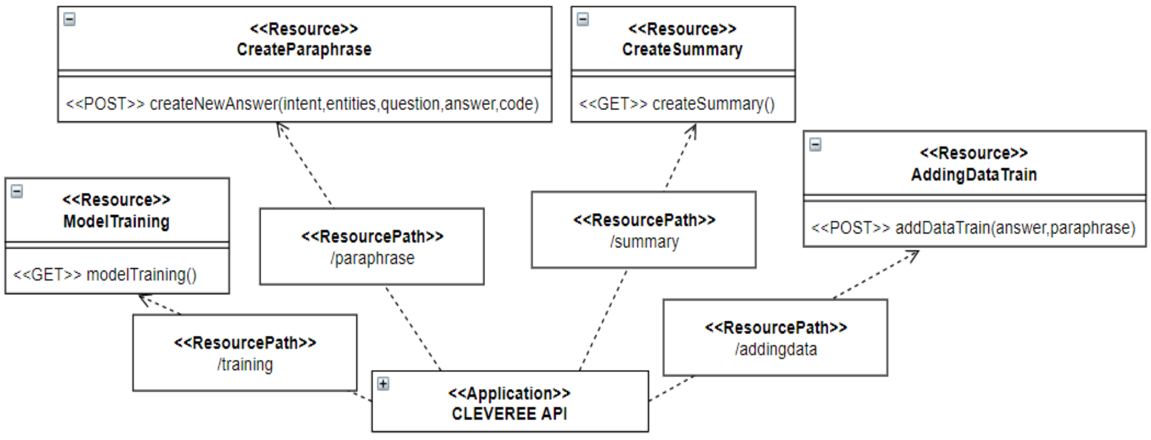
in Formula 2, symbol . at $\vec{i}.\vec{j}$ denotes the dot-product of vector $\vec{i}$ and vector $\vec{j}$. The notation of $||\vec{i}||_2$ represents the vector magnitude of vector $\vec{i}$. In [22], Cosine Similarity is also used to measure the sentence similarity with the Malayalam language.

2.3. Proposed method

2.3.1. Design

The design of Cleveree takes into consideration several Jacob specifications. Jacob uses English as the language and Wit.ai platform to get the intent and entities from the conversation. Jacob's knowledge base is stored using the MySQL database and there is no implementation of any artificial intelligence features in Jacob except for the Wit.ai platform. Jacob is a web-based application built using PHP programming language and Laravel Framework. Jacob could only respond to questions that have answers in the database, furthermore, one question has only one answer. The Wit.ai platform sometimes could give different context or does not recognize the context for similar questions. Therefore, two artificial intelligence features are proposed to be added to Jacob. The purpose is to make Jacob more intelligent by replying to the same or similar questions with a variation of answers. In addition to that is to update the knowledge base into the Wit.ai platform.

The Cleveree is designed as a web service with Python programming language and Flask framework to allow the application to be platform and technology independent. The web service designed with four accessible URLs is shown in Figure 4. The URLs are for paraphrase generation, questions summarization, add training data, and training the model. When the data is sent to the web service, the web service is designed to receive the data using POST method request. Thus, this web service could be accessed by any web application not only Jacob. Additional changes are made in Jacob's knowledge base, it is to add three tables to store the results of paraphrase generation and summarization.



<<Resource>> CreateParaphrase
<<POST>> createNewAnswer(intent,entities,question,answer,code)
<<Resource>> CreateSummary
<<GET>> createSummary()
<<Resource>> AddingDataTrain
<<POST>> addDataTrain(answer,paraphrase)
<<Resource>> ModelTraining
<<GET>> modelTraining()
<<ResourcePath>> /paraphrase
<<ResourcePath>> /summary
<<ResourcePath>> /addingdata
<<ResourcePath>> /training
<<Application>> CLEVEREE API

Figure 4. Cleveree web service model

Cleveree uses a pre-trained model Stacked Residual LSTM for paraphrase generation. In Figure 4, new training data could be added via URL "/addingdata". This action could only be done using an admin role. The admin could choose to train the model with the new dataset to increase the knowledge and the variation of sentences. This process runs in the background so admin can do other activities.

When Jacob calls the URL "/training" and the training process is running, we create a new vocabulary based on the new dataset. After that, the weights in neural networks are always re-initialized. All of the characters are in lowercase and punctuation is not taken into account for the input and output in the model. If the word is not found in the vocabulary database, the word is set to <UNK> to represent an unknown word as in [9]. This aims to learn new words and new sentences. The training process runs in the background so admin can do other activities.

Jacob receives the sentence or question from the user, and sends a request to URL "/paraphrase". Then, starting with lowercase all of the words in the sentence. Next, load the paraphrase model that has been trained before. The model gives the prediction as a result of paraphrase based on the sentence as shown in Figure 5. The prediction is a list of numbers. The number represents the id of word so every word has a different number representation. Then, the number is converted to word and concatenate all of the words. Last, the paraphrased sentence is stored in Jacob's database so the admin can validate and delete the result in the Jacob admin system. If there are errors and admin still understand the context, admin can correct the sentence then validates it so the sentence can be used as an answer. The errors of sentences can be grammatical errors, syntactical errors, and semantic errors.

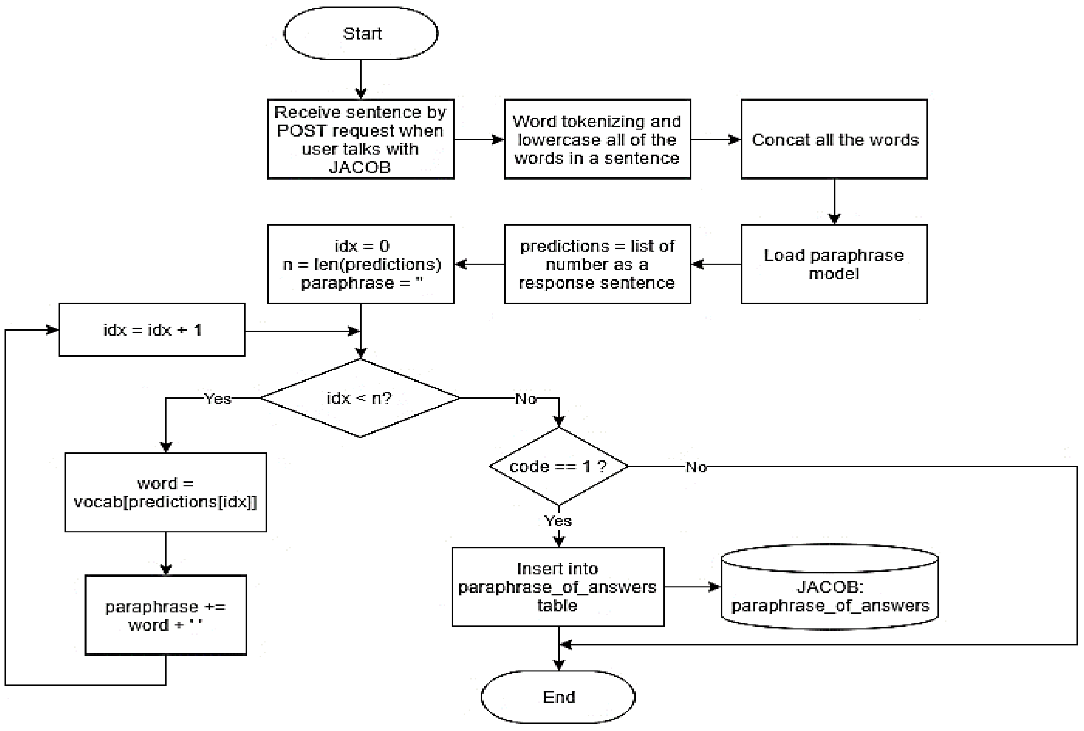

Start
Receive sentence by POST request when user talks with JACOB
Word tokenizing and lowercase all of the words in a sentence
Concat all the words
Load paraphrase model
predictions = list of number as a response sentence
idx = 0
n = len(predictions)
paraphrase = ''
idx = idx + 1
idx < n?
Yes
No
word = vocab[predictions[idx]]
paraphrase += word + ' '
code == 1 ?
No
Yes
Insert into paraphrase_of_answers table
JACOB: paraphrase_of_answers
End

Figure 5. Flowchart of paraphrase generation

There are several steps to do question summarization when Jacob calls the URL "/summary" and it describes in Figure 6. First, get the conversation log files that haven't been summarized. This conversation log files are stored when there is a user interacts with Jacob. Second, do the summary preprocessing. In summary preprocessing, we remove all the punctuations and numbers, tokenize the sentences into words, change all of the words to lowercase, remove stopwords, and load the pre-trained word vectors. Then, calculate the sentence vectors by averaging the total of word vectors for each sentence. Third, create the similarity matrix with size n x n, where n is the total of sentences. The value for each row and column is calculated using the Cosine Similarity method that represents the similarity of each two-sentence vectors. The row in the matrix represents the first sentence and the column represents the second sentence that wants to compare. Fourth, do summary extraction using the TextRank algorithm. The weights in every edge of the graph use the value from the similarity matrix. Then, calculate the scores for every edge using (1) until it reaches the convergence [13]. The last, choose 25% sentences which have the highest score from the entire questions or requests that ask by user [10, 11]. Same as the result of paraphrase, the results of question summarization are stored in the database and admin can validate or delete the result of question summarization.

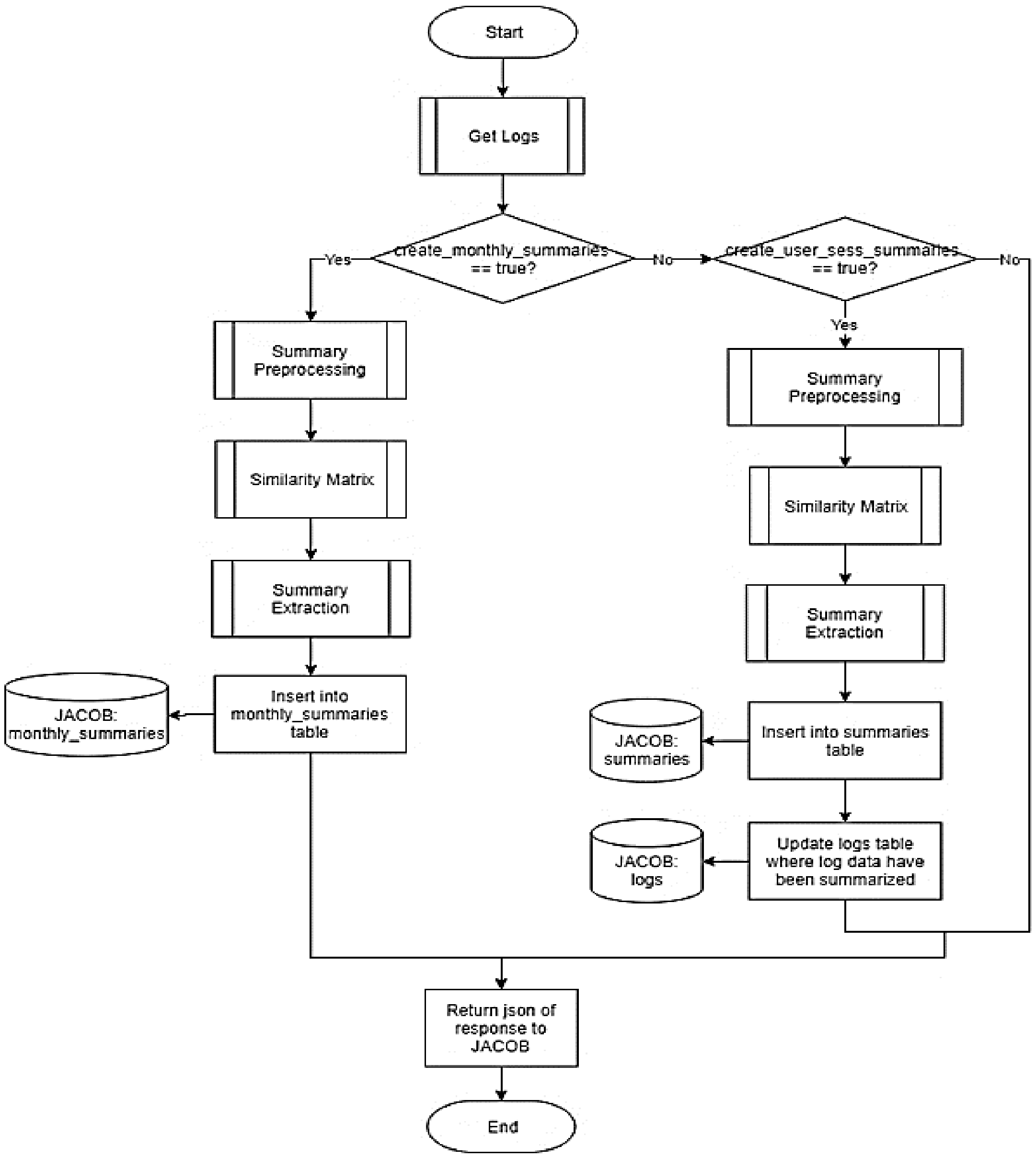

Start
Get Logs
create_monthly_summaries == true?
Yes
No
create_user_sess_summaries == true?
Yes
No
Summary Preprocessing
Similarity Matrix
Summary Extraction
Insert into monthly_summaries table
JACOB: monthly_summaries
Summary Preprocessing
Similarity Matrix
Summary Extraction
Insert into summaries table
JACOB: summaries
Update logs table where log data have been summarized
JACOB: logs
Return json of response to JACOB
End

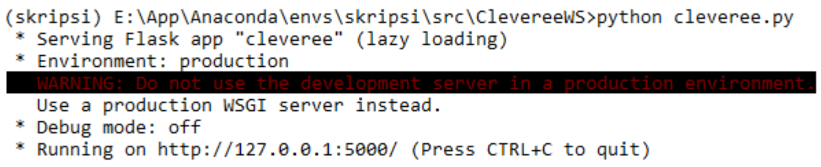
Figure 6. Flowchart of summarization

We use the Whitebox testing approach to measure the implementation of the Cosine Similarity method. It is because this testing is suitable to test the algorithm. Unlike the other research about Artificial Intelligence generally, in this paper, we use the Technology Acceptance Model (TAM) as an evaluation method. TAM is used to predict the acceptance of technology in an organization. The results of TAM are determined based on two perceived variables which are Perceived Usefulness and Perceived Ease of Use [23]. We use the initial scale items for Perceived Usefulness and Perceived Ease of Use [24, 25].

Training data for the Stacked Residual LSTM model is stored in Text Documents, consisting of train source and train target. There is no test dataset for this model, but for the evaluation, we use an interview with Jacob admins to know the paraphrase results are feasible or not. Other than paraphrase results, we do an interview to support the evaluation results using TAM and to know the summarization results are feasible or not. Evaluations use TAM and interview, we do these two evaluations with three admins.

2.3.2. Implementation

This part explains the integration of Cleveree module into Jacob application, the result of the implementation of Stacked Residual LSTM, Cosine Similarity, and TextRank. Cleveree module is built like the description in the research method which is built as a web service and the result is shown in Figure 7.

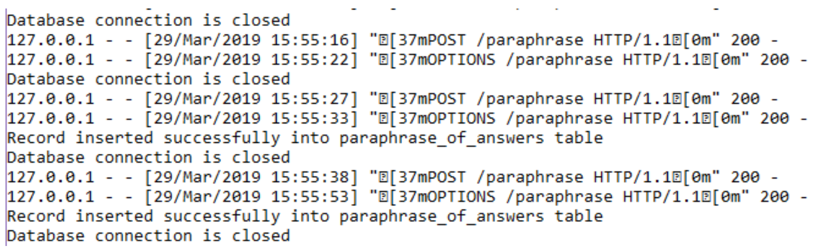


```
(skripsi) E:\App\Anaconda\envs\skripsi\src\CevereeWS>python cleveree.py
 * Serving Flask app "cleveree" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 7. Screenshot of command prompt that shows the Cleveree server is on

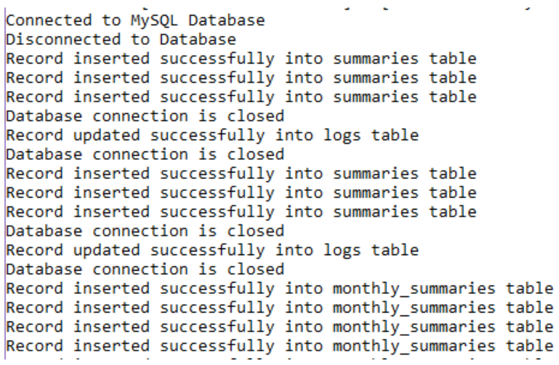
2.3.3. Integration in jacob admin system

The integration in this part means the time when Cleveree is called by Jacob. Jacob calls the URL "/paraphrase" when the admin wants to create a new answer and when Jacob gives the answer to the user that is shown in Figure 8. Then, Jacob calls the URL "/summary" when admin opens the login page which is shown in Figure 9. Jacob calls the URL "/addingdata" when admin creates a new answer so the sentences are added to train source and train target data. Based on the answer that created by admin, Cleveree would create the paraphrased sentence. Then, admins can accept or revise the sentence. Last, Jacob calls the URL "/training" when an admin wants to update the knowledge and variation of answers based on a new dataset.



```
Database connection is closed
127.0.0.1 - - [29/Mar/2019 15:55:16] "□[37mPOST /paraphrase HTTP/1.1□[0m" 200 -
127.0.0.1 - - [29/Mar/2019 15:55:22] "□[37mOPTIONS /paraphrase HTTP/1.1□[0m" 200 -
Database connection is closed
127.0.0.1 - - [29/Mar/2019 15:55:27] "□[37mPOST /paraphrase HTTP/1.1□[0m" 200 -
127.0.0.1 - - [29/Mar/2019 15:55:33] "□[37mOPTIONS /paraphrase HTTP/1.1□[0m" 200 -
Record inserted successfully into paraphrase_of_answers table
Database connection is closed
127.0.0.1 - - [29/Mar/2019 15:55:38] "□[37mPOST /paraphrase HTTP/1.1□[0m" 200 -
127.0.0.1 - - [29/Mar/2019 15:55:53] "□[37mOPTIONS /paraphrase HTTP/1.1□[0m" 200 -
Record inserted successfully into paraphrase_of_answers table
Database connection is closed
```

Figure 8. A case when URL "/paraphrase" is called

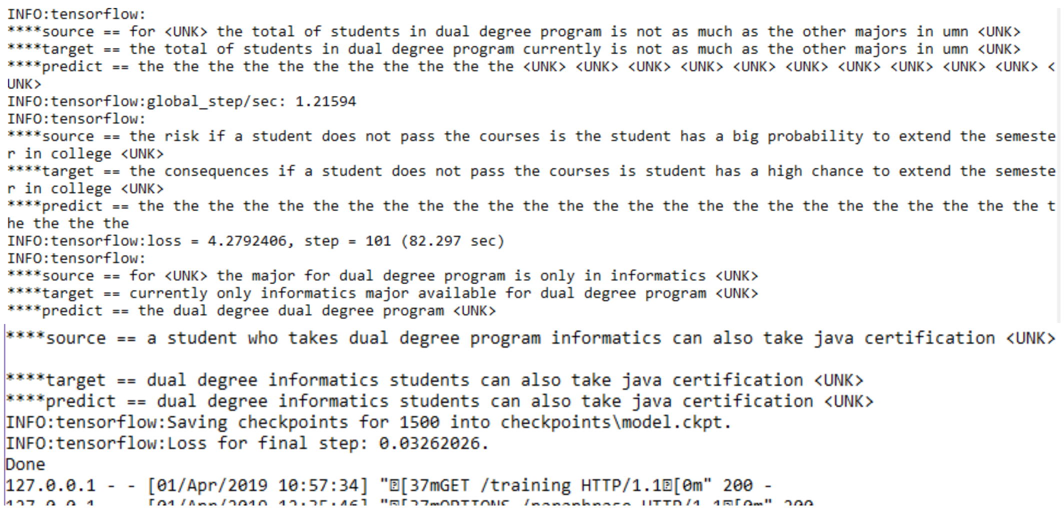


```
Connected to MySQL Database
Disconnected to Database
Record inserted successfully into summaries table
Record inserted successfully into summaries table
Record inserted successfully into summaries table
Database connection is closed
Record updated successfully into logs table
Database connection is closed
Record inserted successfully into summaries table
Record inserted successfully into summaries table
Record inserted successfully into summaries table
Database connection is closed
Record updated successfully into logs table
Database connection is closed
Record inserted successfully into monthly_summaries table
Record inserted successfully into monthly_summaries table
Record inserted successfully into monthly_summaries table
Record inserted successfully into monthly_summaries table
```

Figure 9. The case when URL "/summary" is called

2.3.4. Implementation of pre-trained model of stacked residual LSTM

The implementation of this pre-trained model is using Tensorflow package in Python programming language. The input and output dimensions are the same, the dimension size is 256. The dimension of word embedding is the same as the input and output dimensions. In training, the model uses 0.001 with a fixed value as the learning rate and Adam optimizer. The batch size is set to 32 and the number of iterations is set to 1,500. Every two LSTM layer is added with the residue as in [9]. The total training data is 516 sentences and the total vocabulary is 901 words, but it can increase if admin adds the new sentences of the answer. The training data contains sentences about the Dual Degree program of Informatics in Universitas Multimedia Nusantara and its get from interview with the Marketing Division of Universitas Multimedia Nusantara. Before process the sentence into the model, we need to do preprocessing. In the preprocessing, we use nltk package to remove punctuation and tokenize the sentence to words. Figure 10 represents the indicator when the training process is running. In the training process, we show the source, target, and predict sentence. This process is running in the background for the administrator to carry out other activities.



```
INFO:tensorflow:
****source == for <UNK> the total of students in dual degree program is not as much as the other majors in umn <UNK>
****target == the total of students in dual degree program currently is not as much as the other majors in umn <UNK>
****predict == the the the the the the the the the the the <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <
UNK>
INFO:tensorflow:global_step/sec: 1.21594
INFO:tensorflow:
****source == the risk if a student does not pass the courses is the student has a big probability to extend the semeste
r in college <UNK>
****target == the consequences if a student does not pass the courses is student has a high chance to extend the semeste
r in college <UNK>
****predict == the the the the the the the the the the the the the the the the the the the the the the the the the the t
he the the the
INFO:tensorflow:loss = 4.2792406, step = 101 (82.297 sec)
INFO:tensorflow:
****source == for <UNK> the major for dual degree program is only in informatics <UNK>
****target == currently only informatics major available for dual degree program <UNK>
****predict == the dual degree dual degree program <UNK>
****source == a student who takes dual degree program informatics can also take java certification <UNK>

****target == dual degree informatics students can also take java certification <UNK>
****predict == dual degree informatics students can also take java certification <UNK>
INFO:tensorflow:Saving checkpoints for 1500 into checkpoints\model.ckpt.
INFO:tensorflow:Loss for final step: 0.03262026.
Done
127.0.0.1 - - [01/Apr/2019 10:57:34] "[37mGET /training HTTP/1.1[0m" 200 -
```

Figure 10. Training process of stacked residual LSTM

2.3.5. Implementation of cosine similarity and TextRank

The implementation of the Cosine Similarity method uses numpy package for preprocessing and calculation vectors of each sentence pairs. TextRank algorithm is implemented using networkx package. In preprocessing data of summarization feature is used nltk package to remove stopwords and tokenize sentence to list of words.

3. RESULTS AND ANALYSIS

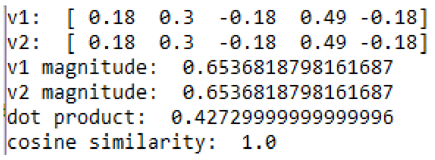
3.1. Testing

3.1.1. Testing for implementation of cosine similarity

Based on the scenario in Table 1, we have two vectors. If we manually calculate using Formula 2, we can get the value of Cosine Similarity is 1 because both vectors exactly have the same direction. In Table 2, we have two different values of vectors and we get the Cosine Similarity score is 0.7774 using Formula 2. Figure 11 and Figure 12 are the results that are given by our module, Cleveree. Thus, we can conclude that the implementation of the Cosine Similarity method in Cleveree is success.

Table 1. First scenario to test the implementation of cosine similarity method

Variable	Condition
Vector dimension	Five
Vector 1	[0,18 0,3 -0,18 0,49 -0,18]
Vector 2	[0,18 0,3 -0,18 0,49 -0,18]
Expected output	The result of the manual calculation for vector magnitude, dot product, and cosine similarity is the same as the result shown in Figure 11.

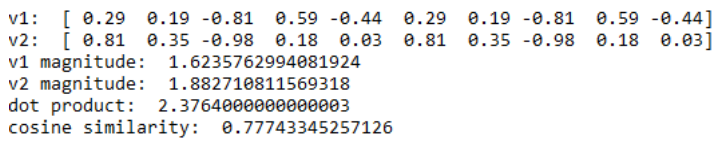


```
v1:  [ 0.18  0.3  -0.18  0.49 -0.18]
v2:  [ 0.18  0.3  -0.18  0.49 -0.18]
v1 magnitude:  0.6536818798161687
v2 magnitude:  0.6536818798161687
dot product:  0.42729999999999996
cosine similarity:  1.0
```

Figure 11. The similarity result using cosine similarity method for the first scenario

Table 2. Second scenario to test the implementation of cosine similarity method

Variable	Condition
Vector dimension	Ten
Vector 1	[0,29 0,19 -0,81 0,59 -0,44 0,29 0,19 -0,81 0,59 -0,44]
Vector 2	[0,81 0,35 -0,98 0,18 0,03 0,81 0,35 -0,98 0,18 0,03]
Expected output	The result of the manual calculation for vector magnitude, dot product, and cosine similarity is the same as the result shown in Figure 12.



```
v1:  [ 0.29  0.19 -0.81  0.59 -0.44  0.29  0.19 -0.81  0.59 -0.44]
v2:  [ 0.81  0.35 -0.98  0.18  0.03  0.81  0.35 -0.98  0.18  0.03]
v1 magnitude:  1.6235762994081924
v2 magnitude:  1.882710811569318
dot product:  2.3764000000000003
cosine similarity:  0.77743345257126
```

Figure 12. The similarity result using cosine similarity method for the second scenario

3.2. Evaluation

The evaluations are carried out using the TAM evaluation method and interview with three Jacob administrators. The TAM evaluation is done using questionnaires to measure the Perceived Usefulness and Perceived Ease of Use. Figure 13 presents the percentage for each question in Perceived Usefulness variables where question number one (Job Difficult Without), three (Job Performance), and fourteen (Usefull for Summary Feature) have the highest score. It is because Cleveree's features can help and increase the performance of the Jacob admin job, especially in the Question Summarization feature. The smallest score is achieved on question number twelve about Makes Job Easier. It is due to the few usage instructions in Jacob administrator pages so sometimes admin feels confused using the admin system. Overall, the total percentage for Perceived Usefulness is 79.17% (strongly agree) that these two features of the Cleveree module are useful.

Figure 14 represents the percentage for each question in Perceived Ease of Use variables where question numbers fourteen (ease to use for summary feature) and fifteen (ease to use for paraphrase feature) have the highest score. It is because both features make the way to update Jacob's knowledge easier. The smallest score is obtained on question number nine about Unexpected Behavior. It is due to the paraphrasing results that do not match with the actual context, thus making the system unpredictable. Other than unexpected behavior, question numbers three (frustrating), four (dependence on manual), and five (mental effort) also have the smallest score because it has a negative meaning. Paraphrasing results sometimes give semantical, syntactical, or grammatical errors so admins must understand the meaning of the sentence first before verifying the results so it causes the frustrating and mental effort to admins. Dependence on Manual also gets the smallest score because as a whole of a system it needs a manual book so admin can learn how to use both features. Overall, the total percentage for Perceived Ease of Use is 72.57% (strongly agree).

Based on the interview results conducted to evaluate the Cleveree in a qualitative way, it is known that the paraphrase generation gives good results. However, some of the paraphrased sentences still require improvement. There is an incomprehensible sentence from the results of paraphrase because a word appears not in the right place (syntactical error). Thus, it is impacting the context and meaning of the whole sentence. This result does not match with the actual context (semantic error). The cause of this problem is where sentences and words are not available in the training dataset. Hence, the model has never learned the sentences and words before. Another result of the question summarization feature successfully gives the desired question by Jacob's administrators and shows the important questions asked by users.

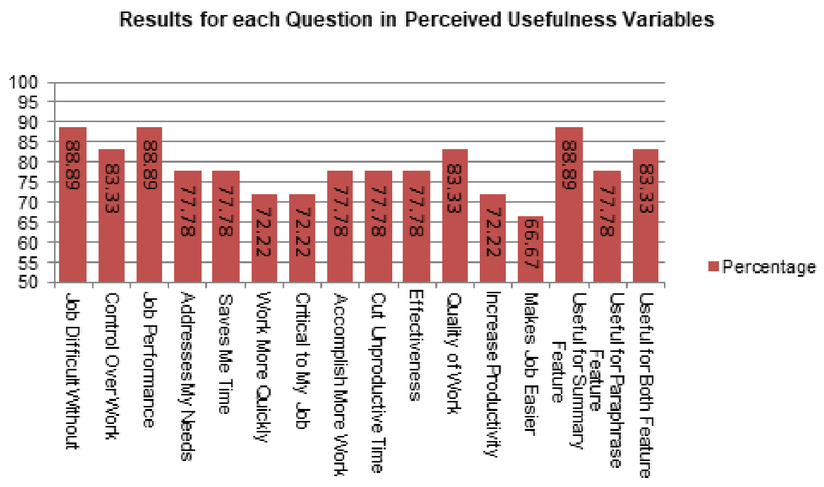


Figure 13. The bar chart for the results of perceived usefulness variables

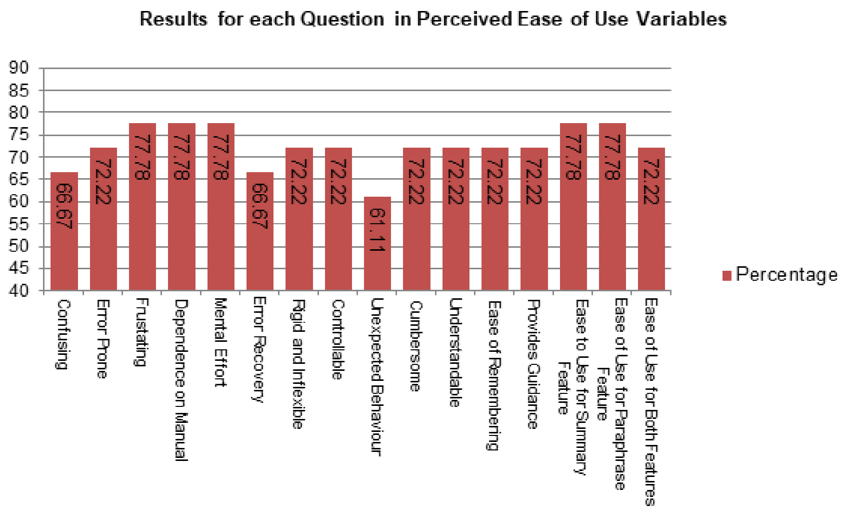


Figure 14. The bar chart for the results of perceived ease of use variables

4. CONCLUSION

We proposed and developed the Cleveree artificial intelligence module for the Jacob voice chatbot application. Cleveree delivers two features: paraphrase of answers and questions summarization. These two features are useful for updating Jacob's knowledge base manually with the help of an administrator. The development of the Cleveree module as a web service eases the integration of the module into Jacob. The pre-trained model of Stacked Residual LSTM is also proven to be successful in generating paraphrase of answers. It is also displayed in this study that the model could be used to generate paraphrase based on the given training dataset. The questions summarization feature powered by the Cosine Similarity method with pre-trained Word2Vec and TextRank algorithm produces satisfactory results as verified by Jacob's administrators. The TAM evaluation method shows that 79.17% of respondents strongly agree that the two features are useful and 72.57% of respondents strongly agree that the two features are easy to use.

REFERENCES

[1] P. A. Angga, et al., "Design of chatbot with 3D avatar, voice interface, and facial expression," *2015 International Conference on Science in Information Technology,* pp. 326-330, 2015.
[2] M. H. Su, et al., "A Chatbot Using LSTM-based Multi-Layer Embedding for Elderly Care," *2017 International Conference on Orange Technologies (ICOT),* pp. 70-74, 2017.
[3] F. P. Putri, H. Meidia, and D. Gunawan, "Designing Intelligent Personalized Chatbot for Hotel Services", *ACAI 2019: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, pp. 468-472, 2019.
[4] S. Wijaya and A. Wicaksana, "Jacob Voice Chatbot Application Using Wit.ai for Providing Information in UMN," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6S3, pp. 105-109, September 2019.
[5] R. Bhagat and E. Hovy, "What Is a Paraphrase?," *Association for Computational Linguistic*. vol. 39, no. 3, pp. 463-472, September 2013.
[6] V. A. Bhagwat, "Deep Learning for Chatbots," M.S. Thesis, Sch. of Comp. Science, San Jose State Univ., San Jose, CA, 2018.
[7] M. Sundermeyer, H. Ney, and R Schlüter, "From Feedforward to Recurrent LSTM Neural Networks for Language Modelling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517-529, 2015.
[8] S. Yavuz, et al., "DEEPCOPY: Grounded Response Generation with Hierarchical Pointer Networks," *32nd Conference on Neural Information Processing Systems, Montréal*, 2019.
[9] A. Prakash, *et al.*, "Neural Paraphrase Generation with Stacked Residual LSTM Networks," *The 26th International Conference on Computational Linguistics*, October 2016.
[10] I. Ferreira, et al., "Bug Report Summarization: An Evaluation of Ranking Techniques," *2016 X Brazilian Symposium on Software Components, Architectures and Reuse*, pp. 101-110, 2016.
[11] S. A. Babar and P. D. Patil, "Improving Performance of Text Summarization," *Procedia Computer Science*, vol. 46, pp. 354-363, 2015.
[12] D. M. Victor, et al., "Application of Extractive Text Summarization Algorithms to Speech-to-Text Media," *International Conference on Hybrid Artificial Intelligence Systems*, pp. 540-550, 2019.
[13] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Text," *Proceedings of the 2004 Conference on Empirical Method in Natural Language Processing*, pp. 404-411, 2004.

[14] 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, pp. 1-7, 2017.
[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
[16] S. J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," 3rd ed. USA: *Pearson Education*, 2016.
[17] A. Mikami, "Long Short-Term Memory Recurrent Neural Network Architectures for Generating Music and Japanese Lyrics," M. S. Thesis, Sch. of Comp. Science, Boston College, Chestnut Hill, MA, 2016.
[18] W. Li and J. Zhao, "TextRank Algorithm by exploiting Wikipedia for short text keywords extraction," *2016 3rd International Conference on Information Science and Control Engineering*, pp. 683-686, 2016.
[19] A. Leeuwenberg, et al, "A Minimally Supervised Approach for Synonym Extraction with Word Embeddings," *The Prague Bulletin of Mathematical Linguistics*, vol. 105, no. 1, pp. 111-142, 2016.
[20] T. Mikolov, et al., "Efficient Estimation of Word Representations in Vector Space," *ICLR: Proceeding of the International Conference on Learning Representations Workshop Track*, Arizona, pp. 1301-3781, 2013.
[21] E. Bigdeli and Z. Bahmani, "Comparing Accuracy of Cosine-based Similarity and Correlation-based Similarity Algorithms in Tourism Recommender Systems," *2008 4th IEEE International Conference on Management of Innovation and Technology*, pp. 469-474, Sep 2008.
[22] P. P. Gokul, B. K. Akhil, and K. K. M. Shiva. "Sentence similarity detection in Malayalam language using cosine similarity," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology*, pp. 221-225, 2017.
[23] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models," *Management Science*, vol. 35, no. 8, pp. 982-1003, Aug 1989.
[24] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *Management Information System Research Center*. pp. 319-340, Sep 1989.
[25] K. Dhammayanti, A. Wicaksana, and S. Hansun, "Position Placement Dss Using Profile Matching and Analytical Hierarchy Process," *International Journal of Scientific & Technology Research,* vol. 8, no. 11, pp. 204-207, Nov 2019.

BIOGRAPHIES OF AUTHORS

Octavany received S. Kom. in Informatics from Universitas Multimedia Nusantara, Indonesia. Her research interests and works are artificial intelligence and software engineering.

Arya Wicaksana is graduated from Universiti Tunku Abdul Rahman in VLSI Engineering (M. Eng. Sc.) and Universitas Multimedia Nusantara in Computer Science (S. Kom.). Research interests and works are: quantum computing and computational intelligence.