ICIC Express Letters
Volume **15**, Number **10**, October **2021**

ICIC International ©2021 ISSN 1881-803X
pp. **1029–1036**

IMPLEMENTATION OF ALBERT FOR TEXT MINING ON JACOB VOICE CHATBOT

Melvin Hendronoto and Arya Wicaksana*

Department of Informatics
Universitas Multimedia Nusantara
Jl. Scientia Boulevard, Gading Serpong, Tangerang 15810, Indonesia
melvin.hendronoto@student.umn.ac.id; *Corresponding author: arya.wicaksana@umn.ac.id

Received January 2021; accepted April 2021

Abstract. *Jacob is a voice chatbot application used to find information about the Multimedia Nusantara University's Informatics Dual Degree program. One of Jacob's shortcomings is the lack of information when answering new questions from users. Therefore, this research focuses on implementing ALBERT (albert-base-v2 and albert-xxlarge-v1 models) as a web service with two modes (explore and explore more) for text mining online information in real time. The implementation is integrated into the Jacob voice chatbot application. The SQuAD 2.0 dataset is used in fine-tuning the applied ALBERT model. The overall accuracy and F-score of the two ALBERT models in this study are 0.58 and 0.7077 for the albert-base-v2 and 0.79 and 0.87 for the albert-xxlarge-v1. The fastest average user time is 1.86579 seconds for the albert-base-v2 in explore mode and the longest is 539.03581 seconds for the albert-xxlarge-v1 in explore more mode.*
Keywords: ALBERT, Text mining, Transformer, Voice chatbot

1. **Introduction.** Artificial intelligence is a popular research field in computer science because it improves people's lives in many fields [1-7]. One of many popular artificial intelligence applications is voice chatbot, and Jacob is one of them [3]. Jacob provides information about the Informatics Dual Degree program at Universitas Multimedia Nusantara (UMN). The answers given by Jacob come from the knowledge database that has stored answers to specific questions. Jacob has a shortcoming in answering new questions due to the limited knowledge base. Fortunately, the Internet today provides almost all the information needed.

Text mining is utilized in this study to retrieve specific information based on the question's keywords. The text mining process consists of searching, fetching, and preprocessing the Internet results in real time. One neural network architecture that is powerful for text mining is ALBERT [8].

ALBERT is developed based on the BERT (Bidirectional Encoder Representation from Transformers) neural network architecture, and it is lighter than BERT. Devlin et al. showed that ALBERT gives the highest F-score of 92.2% for the SQuAD 2.0 dataset, with an increase of 17.4% compared to the BERT neural network architecture [9]. ALBERT's architecture shows a good performance in question answering cases. Zhang et al. obtained an exact match score of 87% and an F-score of 90.2% in machine reading comprehension [10].

This study aims to implement ALBERT and integrate it into the Jacob voice chatbot as a web service. The web service's primary purpose is text mining. The proposed web service is built using the Python 3.5 programming language and the Python Flask 1.0.2 web app framework. Testing and evaluation of the web service are done using the SQuAD 2.0 dataset to measure the accuracy, the F-score, and the user time of the web service.

DOI: 10.24507/icicel.15.10.1029

The rest of this paper is organized as follows. Section 2 briefly describes the preliminaries, including ALBERT, Jacob, and the SQuAD dataset. Section 3 describes the design and implementation of the web service, including the test case. Section 4 presents and discusses the results and evaluation of the implementation in terms of accuracy, F-score, and user time. Section 5 concludes this paper with some discussions on future work.

2. **Preliminaries.** ALBERT (A Lite BERT) is a neural network architecture designed with fewer parameters than BERT architecture [8]. The ALBERT architecture has the same foundation as BERT, namely using encoder transformers and GELU non-linearities. ALBERT uses fewer parameters compared to BERT-large. In the research conducted by El-Geish on the question answering system, he used a stacking ensemble of two models [11].

The study results show that the albert-xxlarge-v1 has the best evaluation results, and the albert-base-v2 has the third-best Top-1 Answer evaluation result, which is not much different from albert-large-v2 even though it has the smallest parameter. Therefore, this study decided to use the albert-xxlarge-v1 and albert-base-v2. This research also decided to take advantage of the fine-tuning of the model in the research conducted by El-Geish. Fine-tuning models are available at https://huggingface.co/models? search = elgeish [11].

Jacob is the voice chatbot application built by Wijaya and Wicaksana in [3] that provides information about the Informatics Dual Degree program in UMN. The general workflow of Jacob voice chatbot is shown in Figure 1 [3].

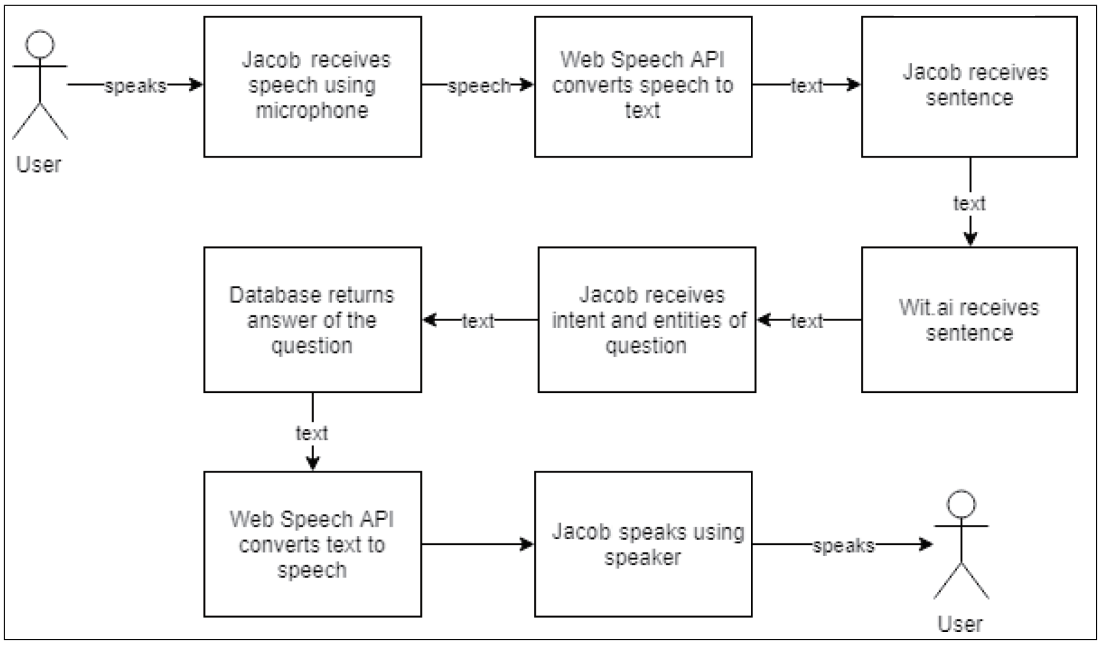

FIGURE 1. Jacob voice chatbot workflow

Jacob was developed as a web application using the PHP programming language with the Laravel framework and Python with the Flask framework. Jacob has four main modules:

- Jacob [3], a module that functions as a voice chatbot, and translates input from the user to obtain the input's essence and identify its name.
- Cleveree [5], a module that functions as a "smart" feature of the voice chatbot, automatic summarization, and answer paraphrasing.
- Vision [7], a module that functions as a "vision" feature, namely in the form of face recognition.
- Virtual Character, a module that functions to display 3D animation of Jacob's character.

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset consisting of more than 100,000 questions on Wikipedia articles. The answers to each question are included in [12]. The SQuAD was created to meet the need for good quality and large datasets. The stages of collecting the dataset were carried out in 3 stages, namely

- Passage curation, collecting the top 10,000 Wikipedia articles to take paragraphs that are considered essential;
- Question-answer collection, asking the crowd workers to give questions and answer five questions from the results obtained in the passage curation;
- Additional answers collection, asking crowd workers to provide at least two additional answers to indicate human performance.

3. **Design and Implementation.** The design process starts with requirement analysis of the web service to allow complete integration with the existing Jacob web app. The only language supported by Jacob is English. Thus, the text mining works only for information in English. Another requirement is to update the question_word entity on Wit.ai used by Jacob to meet the 5W1H and to add the DateTime entity on Wit.ai, which Jacob uses to detect time in user inquiries. The proposed web service model is displayed in Figure 2.

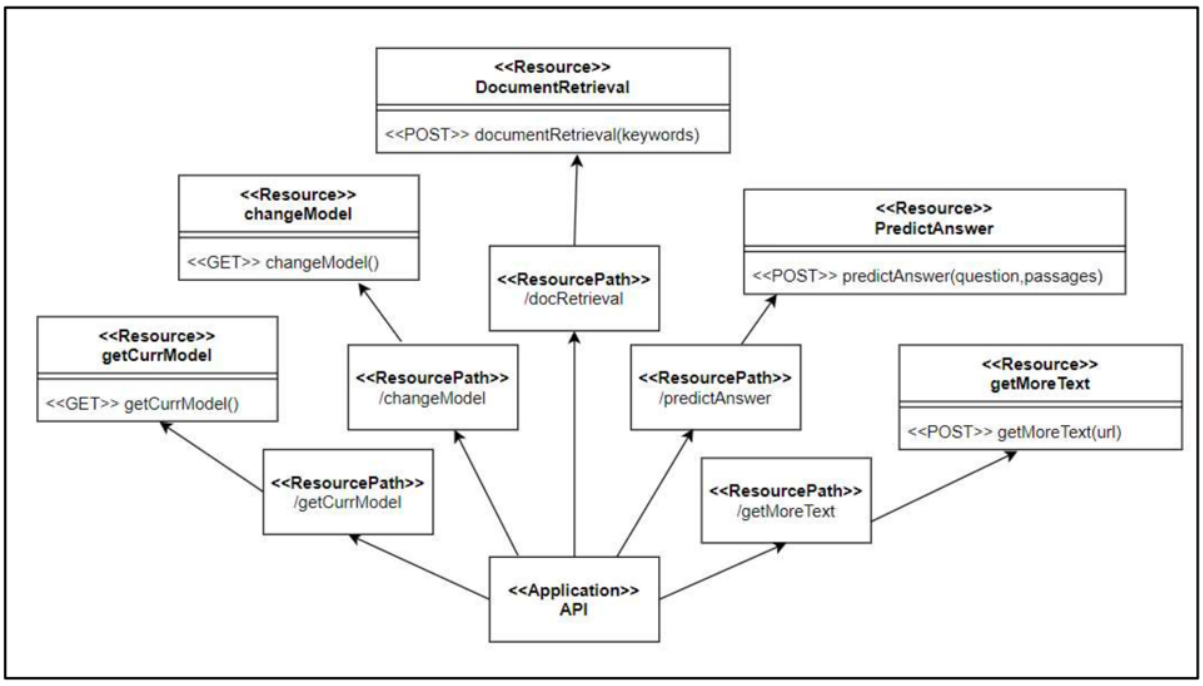

FIGURE 2. Web service model

The web service built has five resources: getCurrModel, changeModel, docRetrieval, predictAnswer, and getMoreText. The getCurrModel is to get the current model used by the web service. The changeModel is to change the model for the web service. The docRetrieval is to get search results using the Google Custom Search API. The getMoreText is to get the text content from the web page. The predictAnswer is to predict the answer to a question from a collection of sentences. The request and response data format is JSON.

Jacob naturally could not answer new questions that have not been recorded in the existing knowledge database. Jacob checks whether or not the question is valid. The checking process uses the question_word entity obtained from wit.ai. If there is a question_word entity, Jacob shows the two buttons' value: the toggle-explore button and the toggle-deeper button. The toggle-explore must be set to true to allow the use of ALBERT.

The toggle-deeper is used to determine whether or not the text mining process fetches more data. The searching step on the Internet is done using the Google custom search engine API. After the text mining process is complete, the web service sends the predicted answer to Jacob. Jacob finally answers the user using the newly retrieved answer from the web service.

Suppose the answer is found from the process. In that case, the variable isFound is set to true, indicating that answers to questions that were not in the database were found and can be entered into the database. To insert into the database, Jacob receives input regarding a confirmation from the user to determine whether the answer is correct or sensible.

In the explore mode, Jacob sends a Document Retrieval post request to the web service to get text search results related to user queries. If the request is successful, Jacob will send a Predict Answer post request to the web service to answer user questions considered in the search result text. In the explore more mode, Jacob requests to post Document Retrieval to the web service. The difference between the two functions lies in the element used from the response obtained when requesting post Document Retrieval, namely the detail element. Jacob will look for answers by retrieving the text obtained from the Get More Text request post with the URL value in detail. The process will be carried out as many as the number of elements in detail. Jacob will take the answer with the highest score (highest probability).

The implementation work consists of interface design implementation, ALBERT implementation, web service integration on Jacob, and Google Custom Search Engine (CSE) API settings. The interface design is implemented by developing the existing Jacob application with the Laravel framework. The interface is made according to the design in the previous chapter. The ALBERT models for predicting answers use pre-trained models from the Transformers library in the Python programming language based on the work of Lan et al. [8]. The model successfully established new state-of-the-art results on the GLUE, RACE, and squad benchmarks while having fewer parameters compared to BERT-large.

The web service implementation and integration to Jacob are done by adding the text "Do you want me to explore the Internet?" and the toggle-explore button with a Yes or No value with the Bootstrap Toggle's help. If the toggle-explore button is Yes, two buttons will appear below it, namely the Change Model button and the toggle-deeper button.

The Change Model button gets the value of the model name currently used by the web service. There are only two values on the Change Model button because the web service only has two models. The toggle-deeper button has two values: Normal and In-depth.

Testing is done to see whether the implementation of ALBERT as a web service can find answers to user questions. The test is categorized into two: the explore mode and the explore more mode. The tests are carried out using 12 test cases shown in Table 1.

4. **Results and Discussion.** In this section, we describe the testing results and the performance evaluation of the two ALBERT models. The performance was evaluated in terms of speed (user time), accuracy, and F-score. We also compare and evaluate the model's performances against the SQuAD dataset. The complete results of the twelve test cases are given in Table 2 for the explore mode and in Table 3 for the explore more mode.

Here we also present the evaluation of the web service according to the test cases in Table 1. We also evaluate the model against the SQuAD dataset with the help of run_squad.py provided in the ALBERT research GitHub repository.

Based on the testing results in Table 2 and Table 3, the XXLarge-v1, in general, gives better answers compared to the Base-v2 model in both explore mode and explore more

mode. The Base-v2 model sometimes could not predict any answer for the given question. The XXLarge-v1 model specifically predicts a better answer than the Base-v2 model, for example, question number eleven, where the model's answer is the CSS scripting syntax. The accuracy and F-score of the model are also given in Figure 3.

Table 1. Test case

Case No.	Question
1	What is an ALBERT NLP?
2	What is Text Mining?
3	Who is Rector of Universitas Multimedia Nusantara?
4	Who is the Founder of Swinburne University?
5	Where will I work?
6	Where do I live?
7	When was Universitas Multimedia Nusantara founded?
8	When was Swinburne Univesity founded?
9	Why to study computer science?
10	Why to choose Universitas Multimedia Nusantara?
11	How to center a div?
12	How much is 1 AUD in IDR?

Table 2. Prediction of answers (explore mode)

Case No.	Model	Answer
1	Base-v2	Not found
	XXLarge-v1	A deep-learning natural language processing model
2	Base-v2	Artificial Intelligence (AI) technology
	XXLarge-v1	An Artificial Intelligence (AI) technology
3	Base-v2	Dr Ninok Leksono
	XXLarge-v1	Dr Ninok Leksono
4	Base-v2	George Swinburne
	XXLarge-v1	George Swinburne
5	Base-v2	Not found
	XXLarge-v1	Not found
6	Base-v2	Ohio
	XXLarge-v1	Not found
7	Base-v2	2005
	XXLarge-v1	2005
8	Base-v2	1908
	XXLarge-v1	1908
9	Base-v2	Having
	XXLarge-v1	Having a computing degree will provide you with the computer technology is at the heart of many endeavors to make a meaningful difference in the world
10	Base-v2	Not found
	XXLarge-v1	The visual communication design study program at universitas multimedia nusantara
11	Base-v2	In a page
	XXLarge-v1	Select
12	Base-v2	9,715.48
	XXLarge-v1	9,266.1474

TABLE 3. Prediction of answers (explore more mode)

Case No.	Model	Answer
1	Base-v2	Not found
	XXLarge-v1	A deep-learning Natural Language Processing (NLP) model
2	Base-v2	Not found
	XXLarge-v1	Text analytics
3	Base-v2	Not found
	XXLarge-v1	Ninok Leksono
4	Base-v2	George Swinburne
	XXLarge-v1	George Swinburne
5	Base-v2	Pharmacy technician
	XXLarge-v1	On or off campus
6	Base-v2	Not found
	XXLarge-v1	We
7	Base-v2	25 November 2005
	XXLarge-v1	25 November 2005
8	Base-v2	1908
	XXLarge-v1	1908
9	Base-v2	If you see yourself designing and creating software systems
	XXLarge-v1	Because it is interesting
10	Base-v2	Not found
	XXLarge-v1	Designed for editing a wide array of media
11	Base-v2	Text-align:center
	XXLarge-v1	Text-align:center
12	Base-v2	10,000
	XXLarge-v1	10,000

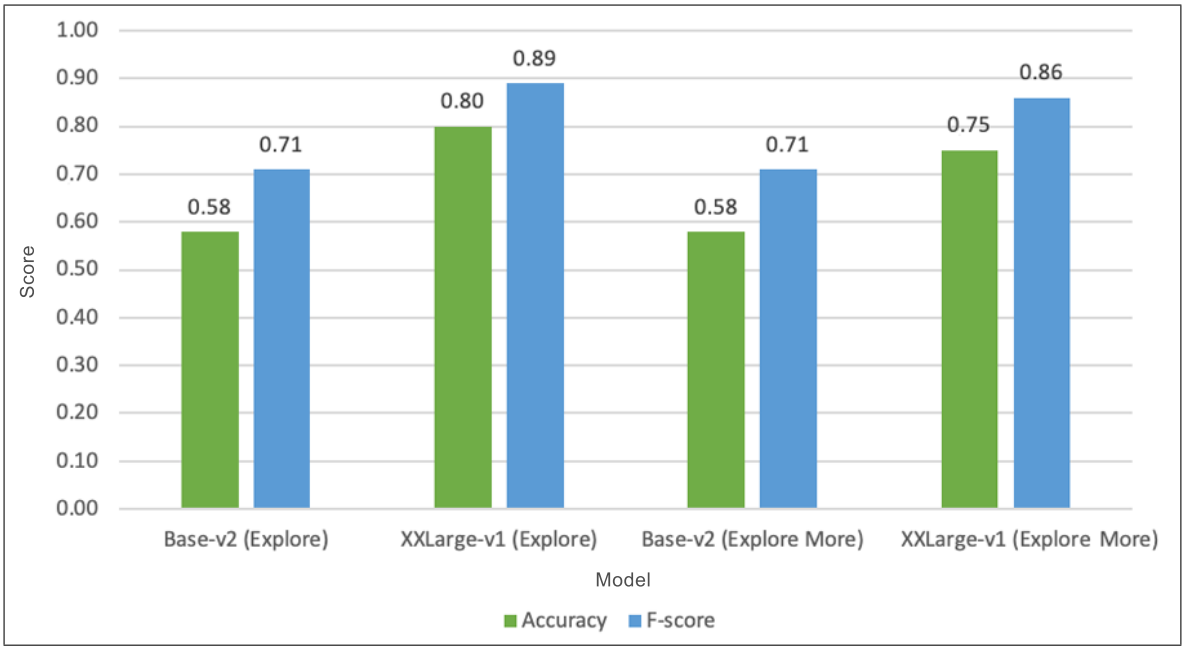

FIGURE 3. Accuracy and F-score of ALBERT implementation

In total, there are four implementations of ALBERT done in this study. The accuracy and F-score presented in Figure 3 show that the XXLarge-v1 with the explore mode delivers the best accuracy and F-score. It is even higher than the XXLarge-v1 model

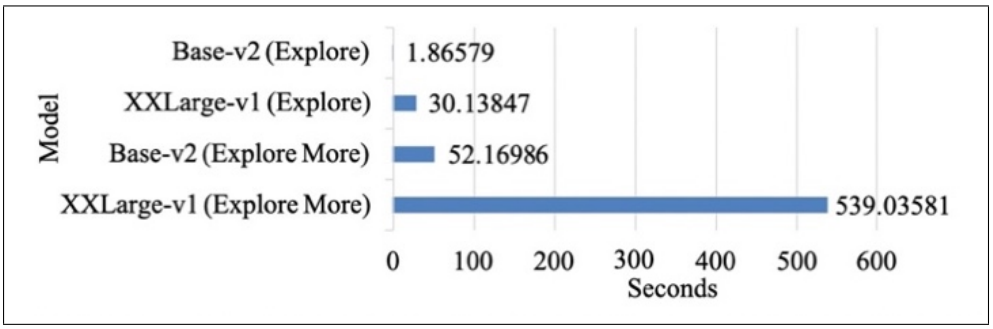

FIGURE 4. Average user time (in seconds)

TABLE 4. User-time comparison (in seconds)

Case No.	Base-v2 (explore)	XXLarge-v1 (explore)	Base-v2 (explore more)	XXLarge-v1 (explore more)
1	1.91188	32.45005	88.61286	638.92878
2	1.71480	28.83286	106.85740	1,353.59997
3	2.00821	29.42678	49.29493	423.79215
4	1.70766	28.31160	33.12582	494.76223
5	1.61744	25.26825	50.48791	605.75582
6	1.65615	25.26825	18.84919	138.52278
7	1.88878	29.03388	56.56785	400.73201
8	1.66120	28.99163	26.44385	419.79450
9	1.58756	26.02800	49.34542	529.10140
10	1.81755	29.64002	42.95223	446.13287
11	2.47170	39.64678	34.03609	457.99803
12	2.34652	38.76354	69.46481	559.30915

TABLE 5. Evaluation against the SQuAD 2.0 dataset

Metric	Base-v2	XXLarge-v1
Accuracy	0.79	0.86
F-score	0.82	0.89

using the explore more mode. The overall accuracy and F-score of the Base-v2 model are 0.58 and 0.71. The overall accuracy and F-score of the XXLarge-v1 are 0.79 and 0.87. Further analysis of the average user time required by each implementation is presented in Figure 4. The recorded user time value is shown in Table 4.

The measured average user time is 1.86579 seconds for the albert-base-v2 and 30.13847 seconds for the albert-xxlarge-v1 in the explore mode, and 52.16986 seconds for the albert-base-v2 and 539.03581 seconds for the albert-xxlarge-v1 in the explore more mode. Based on the results, it is evident that the XXLarge-v1 with the explore mode yields the best performance out of the four implementations.

Further evaluation using the SQuAD dataset is carried out also in this study to evaluate the final model against the SQuAD 2.0. The evaluation is carried out using the Dev-Set dataset provided for evaluation by SQuAD with the help of run_squad.py provided on the ALBERT research GitHub repository. The accuracy and F-score of this test are given in Table 5.

5. **Conclusions.** The web service implementation of ALBERT has successfully done and integrated with the Jacob voice chatbot. The web service addition helps Jacob find

answers to new questions from the Internet in real time. With the options to set the modes: explore and explore more, users could use them as preferred.

The accuracy and F-score of the web service are more than 80% for the ALBERT XXLarge-v1 model implementation. However, the fastest implementation is the ALBERT Base-v2 in explore mode with the shortest average user time of 1.86579 seconds. These numbers show the promising result of the text mining feature in a voice chatbot application. The holy grail is to seek a better model that could not only predict answers more accurately and precisely but also faster.

Based on the research that has been done, future work could be done on fine-tuning the ALBERT even further to achieve higher accuracy and F-score while maintaining the user time as low as possible. The advancement of machine learning algorithms would also enable the implementation of even faster and better neural network models to top the SQuAD leaderboard.

Acknowledgment. This work is fully supported by Universitas Multimedia Nusantara. The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation of the paper.

REFERENCES

[1] K. Dhammayanti, A. Wicaksana and S. Hansun, Position placement DSS using profile matching and analytical hierarchy process, *Int. J. Sci. Technol. Res.*, vol.8, no.11, 2019.

[2] V. Kurniawan, A. Wicaksana and M. I. Prasetiyowati, The implementation of eigenface algorithm for face recognition in attendance system, *Proc. of 2017 the 4th International Conference on New Media Studies (CONMEDIA2017)*, DOI: 10.1109/CONMEDIA.2017.8266042, 2017.

[3] S. Wijaya and A. Wicaksana, JACOB voice chatbot application using wit.ai for providing information in UMN, *Int. J. Eng. Adv. Technol.*, vol.8, no.6 (Special Issue 3), DOI: 10.35940/ijeat.F1017.0986S319, 2019.

[4] Abiyoga, A. Wicaksana and N. M. S. Iswari, Decision support system for choosing an elective course using naive bayes classifier, in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. SNPD 2019. Studies in Computational Intelligence*, R. Lee (edt.), Cham, Springer, 2020.

[5] Octavany and A. Wicaksana, Cleveree: An artificially intelligent web service for Jacob voice chatbot, *Telkomnika (Telecommunication Comput. Electron. Control.)*, DOI: 10.12928/TELKOMNIKA.v18i3.14791, 2020.

[6] K. Alexander, A. Wicaksana and N. M. S. Iswari, Labeling algorithm and fully connected neural network for automated number plate recognition system, in *Applied Computing and Information Technology. ACIT 2019. Studies in Computational Intelligence*, R. Lee (edt.), Cham, Springer, 2020.

[7] A. Archilles and A. Wicaksana, Vision: A web service for face recognition using convolutional network, *Telkomnika (Telecommunication Comput. Electron. Control.)*, DOI: 10.12928/TELKOMNIKA.v18i3.14790, 2020.

[8] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut, ALBERT: A lite BERT for self-supervised learning of language representations, *arXiv.org*, arXiv: 1909.11942, 2020.

[9] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, *arXiv.org*, arXiv: 1810.04805, 2019.

[10] Z. Zhang, J. Yang and H. Zhao, Retrospective reader for machine reading comprehension, *arXiv.org*, arXiv: 2001.09694, 2020.

[11] M. El-Geish, Gestalt: A stacking ensemble for SQuAD2.0, *arXiv.org*, arXiv: 2004.07067, 2020.

[12] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, SQuAD: 100,000+ questions for machine comprehension of text, *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp.2383-2392, DOI: 10.18653/v1/d16-1264, 2016.