International Journal of Innovative
Computing, Information and Control
Volume **18**, Number **6**, December **2022**

ICIC International ©2022 ISSN 1349-4198
pp. **1719–1734**

COMPARISON OF INDOBERT-LITE AND ROBERTA IN TEXT MINING FOR INDONESIAN LANGUAGE QUESTION ANSWERING APPLICATION

Benny Richardson and Arya Wicaksana*

Department of Informatics
Universitas Multimedia Nusantara
Jl. Scientia Boulevard, Tangerang, Banten 15810, Indonesia
benny.richardson@student.umn.ac.id; *Corresponding author: arya.wicaksana@umn.ac.id

Received April 2022; revised July 2022

Abstract. *Jacob is a voice chatbot application that provides information related to the Informatics Joint Degree program at Universitas Multimedia Nusantara. Jacob is currently designed to be able to do question answering and text mining online in real time for the English language. This study aims to find the best model for question answering and text mining for the Indonesian language and integrated with Jacob as proof of concept. The pre-trained models of IndoBERT-lite and RoBERTa are studied and implemented as a web service. The work includes pre-training and fine-tuning the two models with TyDi QA and Indonesian-translated SQuAD datasets. The goal is to find a model that gives answers in the Indonesian language with the highest accuracy and F-score value. The test and evaluation results indicate that the indobert-lite-squad outperforms the rest for Indonesian question answering and text mining applications.*

Keywords: Fine-tuning, IndoBERT-lite, Indonesian language, Question answering, RoBERTa, SQuAD, Text mining, TyDi QA

1. **Introduction.** Chatbots are designed to recognize user input using pattern matching and to access information from the database to provide an appropriate response [1]. Chatbots are often used to assist human work in conveying information, i.e., in virtual learning systems, customer service, etc. The knowledge of the chatbots depends on the information stored in advance for the chatbot to use. Limitations on the availability of the information confine the knowledge of the chatbots. This can be solved by utilizing information from the Internet online. The web has become a large knowledge repository that provides various information in various forms such as text, audio, graphics, video, and multimedia [2]. The information is unlimited and can be accessed, stored, and updated by users worldwide. However, when searching for information online on the Internet, the process is not straightforward. There are several links containing related information that appear. In those links, there is much information in the form of unstructured text that is not easily used directly by chatbots [3].

Jacob chatbot application is chosen as the target application for the proposed approach. Jacob was developed in 2018 with basic conversational (question-answer) capability in English on specific knowledge: joint degree program at Universitas Multimedia Nusantara (UMN) [4]. The language used on the application is English, and the conversational features provided by Jacob for the Indonesian language are not available. This is due to the limited Indonesian language machine learning and text mining models, including Indonesian datasets, compared to English. Multilingual chatbots are also important topics,

DOI: 10.24507/ijicic.18.06.1719

as suggested in [5,6]. In addition, Jacob suffers from the restricted information stored related to the targeted knowledge only. The source code and application are also fully accessible for this work. This makes Jacob an ideal chatbot application to be the target application in this study.

Related work on this topic is a study by Wilie et al. [7], which is a benchmark "IndoNLU". IndoNLU is the first benchmark for the process of training, evaluation, and Indonesian natural language understanding on machine learning with 12 different tasks. Benchmarks are designed to meet machine understanding of Indonesian's formal and informal languages. IndoNLU also introduced the Indonesian language BERT neural network architecture, named IndoBERT, and the ALBERT architecture, named IndoBERT-lite. The results showed that the pre-trained IndoBERT and IndoBERT-lite models had the highest F-score values compared to other pre-trained models [7]. Besides IndoBERT-lite, another neural network architecture is an extension of the BERT architecture with the name RoBERTa. RoBERTa, introduced by Liu et al. [8], is an acronym for the "Robustly Optimized BERT Pretraining Approach", which can even better and even exceed the performance of the post-BERT method. In the study conducted, the results showed that the RoBERTa architecture had the highest F-score with 94.6% in the SQuAD v1.1 dataset and 89.4% in the SQuAD v2.0 compared to the others in the BERT architecture [9] and XLNet [10].

This study's main contribution is finding the best model for Indonesian language question answering and text mining applications. The selected pre-trained models are IndoBERT-lite and RoBERTa, fine-tuned using the Indonesian-translated SQuAD 2.0 dataset and TyDi QA to produce output in the form of answers to user questions. The target application chosen in this study is the Jacob voice chatbot, and the implementation of the models is carried out as web services using the Python 3.6 programming language and the web framework Flask Python 1.0.2. The IndoBERT pre-trained model was not chosen due to the large number of parameters used, and it is impossible to fine-tune it with the existing system specifications. This study uses a pre-trained IndoBERT-lite model that had been fine-tuned using SQuAD translated in Indonesian. The fine-tuned IndoBERT-lite model can be found at the huggingface repository. The fine-tuning is carried out using the pre-trained RoBERTa model found at huggingface repository. Testing is carried out using a white box testing approach to see the accuracy and F-score values from the implementation of IndoBERT-lite and RoBERTa.

The rest of this paper is divided into four sections. Section 2 presents literature related to this study. Section 3 presents the methods, and Section 4 presents the results and analysis. Finally, Section 5 discusses and concludes the work conferred in this paper.

2. **Preliminaries.**

2.1. **Jacob.**

Jacob is a voice-based chatbot application used by the marketing staff of Universitas Multimedia Nusantara to provide information related to the Joint Degree Informatics study program. The information provided includes costs, curriculum, courses, facilities, and careers based on Jacob's knowledge base [4]. Jacob works by receiving input in the form of sound captured using a microphone and providing output in the form of sound with the help of loudspeakers. Input and output on the Jacob are in English. Jacob is developed on a web base using the Wit.ai platform and uses a MySQL database. The Jacob voice chatbot application has five modules implemented in the Jacob feature: the main module, Cleveree, Vision, Lip-Syncing, and Adaptive Learning.

Jacob's main module works by receiving voice input from the user. The input is converted into text using the Web Speech API library in the form of Speech Recognition,

and Jacob sends the data to Wit.ai. Wit.ai is a natural language understanding (NLU) cloud platform developed by Facebook with a focus on finding meaning or meaning in a sentence [11]. Wit.ai works with concepts to model the behavior of chatbots in recognizing several possible conversations that are used. On the Wit.ai Platform, the text obtained is processed by determining the context of the sentence according to user input in the form of intents and entities. The Wit.ai platform returns the result in the form of JSON data. Then, from the results obtained, the data are matched with the database to produce a response corresponding to user input. After successful matching, Jacob responds to the form of a response received back in the form of text to the user. The text in the form of the answer is processed through the Web Speech API with Speech Synthesis, which converts text into sound [4].

2.1.1. *Cleveree module.* The Cleveree module has a function to perform automatic summarization and paraphrasing of answers on the Jacob voice chatbot using the Flask framework in the Python programming language. This module is developed as a web service for Jacob's voice chatbot application. The automatic summarization feature automatically summarizes the input the user provides to Jacob, and the summary is stored in the database. The paraphrasing feature in the answers increases the variety of answers in Jacob's voice chatbot application [12].

2.1.2. *Vision module.* The Vision module has a function to create a facial recognition feature on Jacob's chatbot so that Jacob can recognize the user. This module is developed using the Flask framework in Python. Jacob recognizes three types of users: regular users, admins, and super admins. If Jacob recognizes the face of the user, the user is categorized as an admin or super admin who can add new knowledge to Jacob. The difference is that super admins have additional privileges to add new admins. Meanwhile, if Jacob does not recognize the user's face, the user is categorized as a regular user and can only conduct a question-and-answer interaction with Jacob's chatbot [13].

2.1.3. *Lip-Syncing module.* This module has a function to build a virtual chatbot character Jacob so that Jacob has a lip-sync feature where the mouth of the virtual character Jacob can adjust to the output generated. This module is developed using the C# and Unity programming languages as a platform for bringing up Jacob's virtual character. Jacob's virtual character is developed as a desktop application connected to the web-based Jacob voice chatbot using a bridge. The bridge is created using the PHP programming language. With the bridge, the answers obtained through the web-based Jacob application can be sent to the desktop-based Jacob application with the help of the JSON Encode method [14].

2.1.4. *Adaptive Learning module.* The Adaptive Learning module is a module that functions to help Jacob's chatbot have the ability to do English text mining online. After doing text mining, this module provides the most appropriate answers to the questions given by the user from the information obtained. This module is built as a web service using the ALBERT (A Lite BERT) neural network architecture, developed using Python [15].

2.2. **IndoBERT-lite.** IndoBERT-lite is one of the pre-trained model variants introduced by IndoNLU [7]. IndoBERT-lite is built based on the ALBERT architecture, which continues the BERT model by applying a factorization and weight sharing system to reduce the use of parameters and time [16]. In the training model process, the IndoBERT-lite architecture uses the Indo4B dataset, containing about 4 billion words with 250 million

Indonesian sentences. IndoBERT-lite architecture is tested with other neural network architectures to complete 12 downstream tasks divided into four categories: single-sentence classification, single-sentence sequence-tagging, sentence-pair-classification, and sentence-pair sequence labeling. The results show that the IndoBERT-lite architecture performs well in classification and sequence labeling tasks. Although not as good as the IndoBERT significant architecture, IndoBERT$_{\text{BASE}}$, XLM-R$_{\text{LARGE}}$, and XLM-R$_{\text{BASE}}$, which are the top three with good performance, the performance of the IndoBERT-lite significant architecture is equivalent to the XLM-R$_{\text{BASE}}$ architecture with 16 times the number of parameters less. The comparison of IndoBERT and other models is given in Figure 1.

Model	#Params	#Layers	#Heads	Emb. Size	Hidden Size	FFN Size	Language Type	Pre-train Emb. Type
Scratch	15.1M	6	10	300	300	3072	Mono	-
fastText-cc-id	15.1M	6	10	300	300	3072	Mono	Word Emb.
fastText-indo4b	15.1M	6	10	300	300	3072	Mono	Word Emb.
IndoBERT-lite$_{\text{BASE}}$	11.7M	12	12	128	768	3072	Mono	Contextual
IndoBERT$_{\text{BASE}}$	124.5M	12	12	768	768	3072	Mono	Contextual
IndoBERT-lite$_{\text{LARGE}}$	17.7M	24	16	128	1024	4096	Mono	Contextual
IndoBERT$_{\text{LARGE}}$	335.2M	24	16	1024	1024	4096	Mono	Contextual
mBERT	167.4M	12	12	768	768	3072	Multi	Contextual
XLM-R$_{\text{BASE}}$	278.7M	12	12	768	768	3072	Multi	Contextual
XLM-R$_{\text{LARGE}}$	561.0M	24	16	1024	1024	4096	Multi	Contextual
XLM-MLM$_{\text{LARGE}}$	573.2M	16	16	1280	1280	5120	Multi	Contextual

FIGURE 1. Baseline model results with the best configuration on benchmark [7]

2.3. **RoBERTa.** The robustly optimized BERT approach (RoBERTa) is an extension of the BERT architecture that has been developed with a total dataset of 160GB in the form of English-language corpora so that it can match or exceed the performance of all post-BERT methods [8]. RoBERTa is built based on the study on BERT pre-training replication [9] by measuring hyperparameters' impact, and the training data size used. RoBERTa has two pre-trained models: RoBERTaLARGE and RoBERTaBASE [8].

2.3.1. *Dynamic masking.* The RoBERTa architecture uses the concept of dynamic masking, where a masking pattern is generated every time a sequence is added to the model. This concept performs better than static masking when pre-training for steps and large datasets.

2.3.2. *Removing Next Sentence Prediction (NSP).* Next Sentence Prediction (NSP) is a binary classification loss for predicting whether two segments follow each other in the text. NSP is designed to improve the performance of downstream tasks such as Natural Language Inference [17] which requires reasoning about the relationship between pairs of sentences. In the study by Liu et al. [8], it is found that using only one sentence can reduce the performance of the downstream task. Thus, the RoBERTa architecture uses complete sentences without using NSP loss.

2.3.3. *Training with large batches.* A training process with many mini-batches can improve speed optimization and end-task performance when the learning rate is increased appropriately [18]. The RoBERTa architecture increases the batch size from 256 sequences on the BERT architecture [9] to 8,000 sequences.

2.3.4. *Byte-level Byte-Pair Encoding (BPE).* Byte-Pair Encoding (BPE) is a combination of character-level and word-level that allows for handling large vocabulary commonly used [19]. BPE relies on a pre-tokenizer that splits the training data into words. The RoBERTa architecture implements BPE using bytes [20] rather than Unicode characters as the basis for subword units. Using bytes, it is possible to learn subword vocabulary up to 50,000 units in size and still be able to encode text input without introducing unknown tokens.

In a study by Liu et al. [8], trials are conducted on the architecture of RoBERTa, $\text{BERT}_{\text{LARGE}}$, and $\text{XLNet}_{\text{LARGE}}$. Researchers conducted training on the RoBERTa architecture with the same hyperparameter settings following the large BERT architecture (24-layer, 1024-hidden, 16-heads, 355M parameters). When controlling the training data process, the results show that the RoBERTa architecture has a higher F-score than the $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ architectures. The comparison of the three architectures is given in Figure 2.

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

FIGURE 2. Results of pre-trained data using three architectures [8]

2.4. **SQuAD.** Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset containing more than 100,000 questions in English published in several Wikipedia articles, where there are answers to each question in the article [21]. The dataset is collected through 3 stages:

- Passage curation. The passage curation stage aims to get the top 10,000 articles on the English Wikipedia and make a random sample of 536 articles;
- Question-answer collection. The question-answer collection stage aims to collect questions and answers made by crowd workers. Each crowd worker is assigned to make as many as five questions from the results of the articles obtained at the passage curation stage;
- Additional answer collection. The additional answer collection stage aims to get an indication of human performance in the dataset by adding at least two answers to each question.

SQuAD has 2 versions: SQuAD v1.1 and SQuAD v2.0 [22]. Version 1.1 of SQuAD has a weakness where the answer to the question is only based on the paragraph content, so it ignores the question outside the paragraph content. In SQuAD v2.0, development is carried out by increasing the amount of data. The dataset contains an amalgamation of SQuAD v1.1 with an additional 50,000 unanswerable questions created by crowd workers to be similar to the answerable questions.

Muis and Purwarianti [23] researched sequence-to-sequence learning to build an Indonesian automatic question generator (AQG). The researchers built the SQuAD v2.0 dataset in Indonesian. Researchers use the dataset as knowledge to conduct training on the system so that machines can understand Indonesian. The dataset is built using the Google Translate API v2 to translate the SQuAD v2.0 dataset from English to Indonesian. The translation results are 536 articles with 161,550 question-answer pairs.

2.5. **Typologically Diverse Languages Question Answering (TyDi QA).** TyDi QA is a question answering dataset that has 11 diverse languages with 204,000 question-answers pairs [24]. The languages spoken include Arabic, Bengali, Finnish (Finno-Ugric), Japanese, Indonesian, English, Kiswahili, Korean, Russian, Telugu, and Thai. Sources of data in the TyDi QA dataset are collected from Wikipedia articles through 3 procedures [24]:

- Question elicitation. Question elicitation is the stage where human annotators are given instructions consisting of the first 100 characters of each Wikipedia article. Then they are asked to write questions for which answers are available and those are not provided in the guide;
- Article retrieval. Article retrieval is the stage to perform a search on Google search by entering questions from human annotators. Searches are made on Wikipedia articles with adjusted language setting restrictions. Then choose the top article from the search performed;
- Answer labeling. Answer labeling is the last stage to choose the best passage answer from the question/article pair. The best passage answer is determined from the presence or absence of an appropriate answer from the paragraph of the article.

TyDi QA dataset has two tasks: primary tasks and secondary tasks [24].

- Primary Task
  - Passage Selection Task (SelectP). Provide a list of passages in the article, then return one of the index passages that answer the question or return "NULL" if there is no passage that answers the question.
  - Minimal Answer Span Task (MinSpan). Provide the full text of an article, and then return one of the starting and ending byte indexes of the minimal span that answers the question, "YES" or "NO" if the question requires a yes/no answer, and "NULL" if it is not possible to produce an answer on the question.
- Secondary Task
  - Golden Passage Task (GoldP). Provide passage containing answers, predicting a single contiguous span of words that answer the question.

3. Methods.

3.1. **Requirement analysis.** Based on the results of the analysis of the Jacob voice chatbot application that previous researchers have built, it is known that

- The language used as input and output on Jacob's chatbot application is English;
- The Jacob voice chatbot application uses one app by using English on the Wit.ai platform to get the value of the intent and entities of a sentence;
- Searching for information online is done using the help of Google Custom Search Engine API, which has English settings so that the documents obtained for the text mining process are based on English;
- Prediction of answers from information obtained online is made using the pre-trained ALBERT model with the help of the Transformers library in the Python programming language;

- Jacob's chatbot application is built using the PHP programming language with the Laravel framework and also uses the Python programming language for web services using the Flask framework;
- The table in the Jacob database contains only a table to store responses to be given in English.

The addition and expansion of the Jacob chatbot and web service implementation to be carried out in this study are as follows.

- Add the Indonesian language as the new language to make Jacob bilingual. This is achieved by providing the Indonesian language as an option for the user.
- Add a new app on the Wit.ai platform for the Indonesian language feature to allow Jacob to receive intents and entities in Indonesian.
- Add Google Custom Search Engine API with Indonesian language setting for searching Indonesian text documents in the text mining process.
- Use the fine-tuned IndoBERT-lite and RoBERTa models with the help of the Transformers and Simpletransformers libraries in the Python programming language to predict answers from the results obtained under reasonable time delay.
- Add a new table to the Jacob database to store responses in Indonesian language.

3.2. **Design.** Based on the analysis described in the previous subchapter, the web service and chatbot application design in the Indonesian language are carried out. In this study, the models used to get the answers for Jacob's users are pre-trained and fine-tuned for the Indonesian language. This process aims to find the best model for the application. The best fine-tuned model is implemented on the web service and integrated with the Jacob voice chatbot application to test and evaluate the approach proposed in this study. Accuracy and F-score are the performance metrics chosen in this study. The development process used in this study is presented in Figure 3. In previous studies and related works, the research on Indonesian-based machine learning models is focused on the technical aspect of the models' algorithm and dataset.

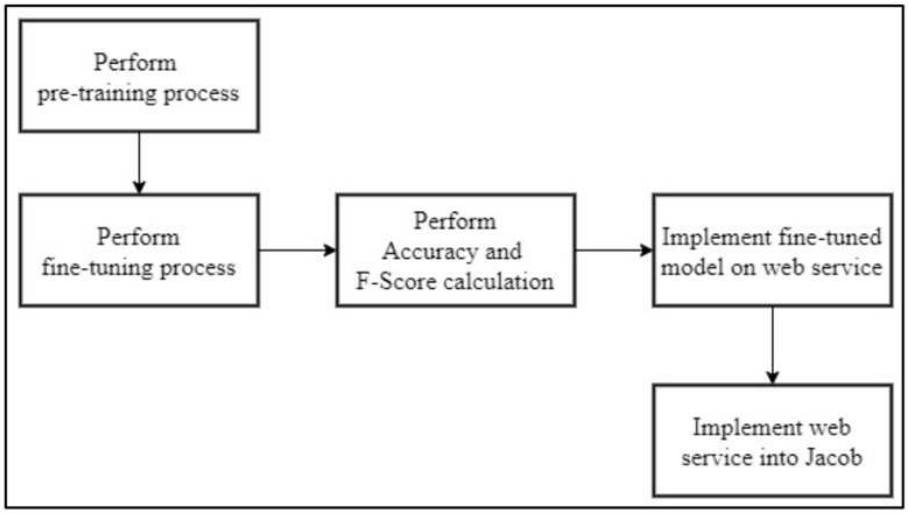

FIGURE 3. Development process

The pre-training is done to get a pre-trained RoBERTa model for the Indonesian language. The dataset used for the pre-training process is the Indo4B dataset. Based on the Indo4B dataset, a tokenizer is made to obtain a list of vocabularies and word pairs in the Indonesian language, which is required in the training process for the model. The training dataset is created from the tokenizer. The produced model is used for masking

but cannot yet deliver Question Answering (QA) capability. The fine-tuning process of this model is carried out using the QA dataset.

The contribution of this study that expands previous work on Jacob web services lies in how the pre-trained model works to provide predictive answers to the user in the Indonesian language. The web service works begin with the Jacob voice chatbot application sending input in the form of questions from the user to the web service with a POST request. The web service receives questions from users and uses them as keywords to search for information online. The search results are obtained in several documents containing information related to user questions. Jacob receives the data and processes the data. After processing, Jacob sends the data back via a POST request. The web service receives the data that Jacob has processed. The web service uses the data through a pre-trained model applied to predicting answers according to user questions. The predicted answers from the pre-trained model are stored in JSON form and sent to Jacob.

3.3. **Implementation.** The implementation work is carried out in Google Colaboratory (Colab). The specifications of the Google Colab used are as follows.

- Graphical Processing Units (GPU):
  - 12.69GB RAM
  - 68.40GB Disk Space
  - NVIDIA Tesla K80
- Tensor Processing Unit (TPU):
  - 25GB RAM
  - 128GB Disk Space
  - NVIDIA P100

The pre-trained RoBERTA model is implemented by conducting the pre-trained model process, the model fine-tuning process, and the application of the pre-trained RoBERTA model on the web service.

3.3.1. *Pre-trained model.* The process begins with training the tokenizer using the assistance from ByteLevelBPETokenizer and the Indo4B dataset. The results of the tokenizer obtained are in the form of two files, namely the vocab.json and merges.txt files. The vocab.json file contains a collection of Indonesian vocabulary, while the merges.txt file contains the dataset's most frequently used byte pairs. The time needed to carry out the tokenizer training process on the Indo4B dataset is 2 hours 25 minutes. The configuration carried out follows the hyperparameters on the RoBERTa-based architecture. These include the vocab_size of 52,000, the max position of 514, attention heads of 12, a hidden layer of 6, and other configurations in the RobertaForMaskedLM library. The DataCollatorForLanguageModeling library organizes the obtained training datasets into batches as a dictionary of tensors. The process is carried out using the help of a trainer with a batch size configuration of 16 per device.

3.3.2. *Fine-tuned model.* The fine-tuning process uses a pre-trained model obtained from the pre-training process and the QA dataset. The QA dataset used is the Indonesian SQuAD dataset and the TyDi QA dataset. The fine-tuning process is carried out using assistance from the Simpletransformers library and the FARM (Framework for Adapting Representation Models) library. The FARM library is used to fine-tune the SQuAD dataset, while the Simpletransformers library is used to fine-tune the TyDi QA dataset.

Fine-tuning begins with data processing by creating a tokenizer from the pre-trained model. After getting the tokenizer, create a data processor using the SQuAD dataset. The data processor converts the dataset from raw text to the Pytorch dataset. The next step is to fine-tune the model. The fine-tuning process is carried out using hyperparameter

settings such as the batch size of 16, the learning rate of 1e-5, the epoch of 2, and other settings in the library.

The hyperparDataSilo loads datasets in the train, test, and valid tests. The fine-tuning process is carried out with the help of a trainer. The data preprocessing process is initially carried out by fine-tuning the model using the TyDi QA dataset and the help of the Simpletransformers library. Data preprocessing is carried out to ensure that the dataset used in the fine-tuning process is a dataset that only uses Indonesian. The number of epochs used is five in the configuration, and the batch size is 16.

3.4. **Testing and evaluation.** Tests are carried out to see the accuracy and F-score values generated from the pre-training and fine-tuning carried out and to see the testing of the implementation of the IndoBERT-lite and RoBERTa pre-trained models in predicting the answers to user questions. The test scenarios carried out in this study are as follows.

1) Testing the pre-trained model to get the pre-trained RoBERTa model for Indonesian language using the Indo4B dataset.
2) Testing the fine-tuned model to get the fine-tuned RoBERTa model that can perform question-answering in Indonesian language. Tests are carried out using the Indonesian SQuAD and the TyDi QA datasets.
3) Tests on the fine-tuned model obtained through the fine-tuning process in providing predictive answers from the information obtained online. The tests also used 5 fine-tuned models, namely indobert-lite-squad, Roberta-1.5gb-tydiqa, Roberta-3gb-tydiqa, Roberta-1.5gb-squad, and Roberta-3gb-squad. The five fine-tuned models are tested to predict the answers to the 15 questions shown in Table 1.

Table 1. List of questions for the testing

No.	Question in Indonesian	Translation in English
1	*Apa itu text mining?*	What is text mining?
2	*Apa itu machine learning?*	What is machine learning?
3	*Siapa CEO dari perusahaan Google saat ini?*	Who is the CEO of the current Google company?
4	*Siapa pendiri Kompas Gramedia?*	Who is the founder of Kompas Gramedia?
5	*Dimana tempat tinggal saya?*	Where do I live?
6	*Dimana alamat Universitas Multimedia Nusantara?*	What is the address for Multimedia Nusantara University?
7	*Tanggal berapa Kompas Gramedia berdiri?*	What date was Kompas Gramedia established?
8	*Berapa harga saham Bitcoin saat ini?*	What is the current Bitcoin price?
9	*Kapan universitas multimedia didirikan?*	When was the multimedia university founded?
10	*Kapan saya lulus dari universitas multimedia nusantara?*	When will I graduate from Nusantara Multimedia University?
11	*Mengapa diperlukan adanya Artificial Intelligence?*	Why is Artificial Intelligence needed?
12	*Mengapa harga saham turun?*	Why did the stock price drop?
13	*Bagaimana keadaan cuaca saat ini?*	How is the weather right now?
14	*Berapa jam lamanya penerbangan dari Jakarta ke Melbourne?*	How many hours is the flight from Jakarta to Melbourne?
15	*Bagaimana saya dapat lulus?*	How can I graduate?

Based on the test results obtained, an evaluation of the pre-trained models and fine-tuned models is carried out, and an evaluation of the tests of the fine-tuned models is used.

4. **Results and Analysis.** Tests on pre-training are carried out in several experiments using variations in the size of different training data and changing the hyperparameter configuration to train the model. The results of the pre-training process carried out can be seen in Table 2.

Table 2. RoBERTa pre-training results

Dataset	Dataset size	Vocab size	Max steps	Learning rate	Weight decay	Adam epsilon
Wikipedia	500MB	52,000	72,500	1e-12	0.01	1e-6
wiki	363MB	50,265	48,504	5e-5	0.01	1e-6
conllu_all_uncased.txt	6GB	52,000	7,727,763	1e-12	0.01	1e-6
Wikipedia_conlu + opensubtitles	1.5GB	52,000	125,000	1e-5	0.01	1e-6

Dataset	Batch size	Hidden layer	Attention heads	Epoch	Training duration	Accuracy
Wikipedia	16	6	12	1	8h28mins (finished)	0.041
wiki	16	12	12	1	29mins	0.032
conllu_all_uncased.txt	16	6	12	1	3h28mins (loss)	–
Wikipedia_conlu + opensubtitles	16	6	12	1	9h32mins	–

In Table 2, it can be seen that pre-training using a dataset of 500MB in size, the learning rate of 1e-12 has an accuracy value of 0.0408898 with the duration of the pre-training process as 8 hours 28 minutes, and has a max step of 72,500. In a 363MB dataset, a learning rate of 5e-5 has an accuracy value of 0.0322657 with a duration of 29 minutes for the pre-training process and has a max step of 48,504. In the pre-training process using a 6GB dataset and a learning rate of 1e-12, the pre-training process has a max step of 0.0408898 with a duration of 8 hours and 28 minutes for the pre-training process and has a max step of 72,500. In a 363MB dataset, a learning rate of 5e-5 has an accuracy value of 0.0322657 with a duration of 29 minutes for the pre-training process and has a max step of 48,504. In the pre-training process that uses a dataset of 6GB and a learning rate of 1e-12, the pre-training process has a max step of 7,727,763 but stops with a duration of 3 hours because the number of disks used exceeds the disk capacity used for the pre-training process. Meanwhile, in the pre-training process using a 1.5GB dataset and a learning rate of 1e-5, the pre-training process has a max step of 125,000. However, it stops with a duration of 9 hours because the session used in the pre-training process approached the timeout set by Google Colab.

Based on the test results obtained in the pre-training process, it can be seen that the pre-trained model using a dataset below 1.5GB is successfully carried out, but the resulting accuracy is minimal. The model that uses the Wikipedia dataset with 500MB has an accuracy value of 0.041. While the pre-trained model using the Wikipedia dataset with a size of 363MB has an accuracy value of 0.032. This affects the performance resulting from the fine-tuning process using the pre-trained model, where the fine-tuned model has poor

performance when answering the question. In addition, it can be seen that the RoBERTa architecture has better performance if the dataset used in the pre-training has a larger size. The RoBERTa architecture uses a dataset size of 160GB. Thus, testing cannot be carried out using a dataset with a larger size due to the limitations of the Google Colab specification used.

Testing the fine-tuned model process using two datasets, namely the SQuAD and TyDi QA datasets. The results of the fine-tuning process for the RoBERTa model with the SQuAD dataset can be seen in Table 3.

Table 3. RoBERTa fine-tuning results with SQuAD

Dataset	Model	Learning rate	Weight decay	Batch size	Epoch
Squad translated	cahya/Roberta-base-indonesian-1.5G	1e-5	0.01	16	1
Squad transalted	w11wo/indo-roberta-small (3.1GB)	1e-5	0.01	16	1

Dataset	Model	Trial Phase 1		
		EM	F1	Time
Squad translated	cahya/Roberta-base-indonesian-1.5G	0.545	0.594	2h26m43s
Squad transalted	w11wo/indo-roberta-small (3.1GB)	0.456	0.467	1h23m46s

Dataset	Model	Trial Phase 2		
		EM	F1	Time
Squad translated	cahya/Roberta-base-indonesian-1.5G	0.540	0.585	1h21m20s
Squad translated	w11wo/indo-roberta-small (3.1GB)	0.474	0.479	1h18m6s

Dataset	Model	Trial Phase 3		
		EM	F1	Time
Squad translated	cahya/Roberta-base-indonesian-1.5G	0.542	0.590	4h31m43s
Squad translated	w11wo/indo-roberta-small (3.1GB)	0.471	0.477	1h18m13s

Table 3 shows the results of the tests carried out on two RoBERTa models. Testing is done by fine-tuning the model 3 times. The hyperparameters used are learning rate of 1e-5, weight decay of 0.01, batch size of 16, and epoch of 1 time, which values are obtained from the fine-tuning process. This fine-tuning configuration is set up based on the work of RoBERTa and customized accordingly to suit the implementation resources available for this study.

In the first stage, the pre-trained cahya/Roberta-based-Indonesian-1.5G model resulted in an exact match value of 0.545, F1 (F-score) of 0.594, and a training duration of 2 hours 26 minutes. While the pre-trained model w11wo/indo-roberta-small has the results in the form of an exact match value of 0.456, F1 of 0.467, and a training duration of 1 hour 23 minutes. In the second stage, the pre-trained cahya/Roberta-base-indonesian-1.5G model

resulted in an exact match value of 0.540, F1 of 0.585, and a training duration of 1 hour 21 minutes. While the pre-trained model w11wo/indo-roberta-small has the results in the form of an exact match value of 0.474, F1 of 0.479, and a training duration of 1 hour 18 minutes. In the third stage, the pre-trained cahya/Roberta-base-indonesian-1.5G model resulted in an exact match value of 0.542, F1 of 0.590, and a training duration of 4 hours 31 minutes. While the pre-trained model w11wo/indo-roberta-small has the results in the form of an exact match value of 0.471, F1 of 0.477, and a training duration of 1 hour 18 minutes.

The TyDi QA dataset has been designed so that only Indonesian articles are used through preprocessing. The results of the fine-tuning process on the RoBERTa model with the TyDi QA dataset can be seen in Table 4.

TABLE 4. RoBERTa fine-tuning results with TyDi QA

Dataset	Model	Learning rate	Batch size	Epoch
TyDi QA	cahya/Roberta-base-indonesian-1.5G	1e-5	16	5
TyDi QA	w11wo/indo-roberta-small (3.1GB)	1e-5	16	5

Dataset	Model	Trial Phase 1			
		Correct	Incorrect	Similar	Time
TyDi QA	cahya/Roberta-base-indonesian-1.5G	153	99	313	17m6s
TyDi QA	w11wo/indo-roberta-small (3.1GB)	112	163	290	9m5s

Dataset	Model	Trial Phase 2			
		Correct	Incorrect	Similar	Time
TyDi QA	cahya/Roberta-base-indonesian-1.5G	154	96	315	20m10s
TyDi QA	w11wo/indo-roberta-small (3.1GB)	112	156	297	9m15s

Dataset	Model	Trial Phase 3			
		Correct	Incorrect	Similar	Time
TyDi QA	cahya/Roberta-base-indonesian-1.5G	156	97	312	16m49s
TyDi QA	w11wo/indo-roberta-small (3.1GB)	112	155	298	9m13s

Table 4 shows the results of the tests carried out by fine-tuning the model 3 times. The hyperparameters used are learning rate of 1e-5, batch size of 16, and epochs of 5 times. The test results obtained are correct, incorrect, and similar values. Correct is the number of predicted answers that match the correct answer. Similar is the number of predicted answers which are a substring of the correct answers. Incorrect is the number of predicted answers with no correct and similar criteria.

In the first phase, the pre-trained model of cahya/Roberta-base-indonesian-1.5G resulted in 153 correct, 99 incorrect, and 313 similar answers with 17 minutes of training duration. While the pre-trained model w11wo/indo-roberta-small has 112 correct, 163 incorrect, and 290 similar answers with 9 minutes training duration. In the second phase, the pre-trained model cahya/Roberta-base-indonesian-1.5G resulted in 154 correct, 96 incorrect, and 315 similar answers with 20 minutes training duration. While the pre-trained model w11wo/indo-roberta-small resulted in 112 correct, 156 incorrect, and 297 similar answers with 9 minutes training duration. In the third phase, the pre-trained model cahya/Roberta-base-indonesian-1.5G resulted in 156 correct, 97 incorrect, and 312 similar answers with 16 minutes of training duration. At the same time, the pre-trained model

w11wo/indo-roberta-small resulted in 112 correct, 155 incorrect, and 298 similar answers with 9 minutes of training duration.

Based on the results of the tests carried out in the fine-tuning process, it can be seen in the SQuAD dataset that the Indonesian translation of the fine-tuned model has Exact Match (EM) and F-score values. The pre-trained model of cahya/Roberta-base-indonesian-1.5G has an average EM value of 0.542 and an F-score of 0.589. The pre-trained model w11wo/indo-roberta-small has an average EM value of 0.467 and an F-score of 0.474. In the TyDi QA dataset, the results of fine-tuning the model obtained are correct, incorrect, and similar values. The pre-trained model cahya/Roberta-base-indonesian-1.5G has an average correct value of 154, incorrect of 97, and similar of 313. In contrast, the pre-trained model w11wo/indo-roberta-small has an average correct value of 112, incorrect of 158, and similar of 295. The performance generated from each fine-tuned model has quite good results, which can be used in the question answering system.

Based on the two results obtained, there is an anomaly where the pre-trained model w11wo-indo-roberta-small should have higher accuracy, F-score, and correct value than the pre-trained model cahya-roberta-base-Indonesian-1.5G. This is because the pre-trained model w11wo-indo-roberta-small uses a dataset of 3GB. This can be caused by a system where the use of hyperparameters set in the pre-trained model cahya/Roberta-base-indonesian-1.5G is greater than the hyperparameters set during the pre-trained model process in the pre-trained model w11wo-indo-roberta-small.

The final evaluation measures the accuracy and F-score obtained from the five fine-tuned models when predicting answers. Table 5 shows the confusion matrix results from the tests carried out on 15 questions. The graph regarding the accuracy and F-score values from the five pre-trained models tested for answer prediction can be seen in Figure 4.

Table 5. Confusion matrix results

Model	TP	TN	FP	FN
indobert-lite-squad	12	0	0	3
Roberta-1.5gb-tydiqa	10	2	2	1
Roberta-3gb-tydiqa	5	2	7	1
Roberta-1.5gb-squad	7	0	5	3
Roberta-3gb-squad	6	0	6	3

Based on the results, the five fine-tuned models have an average F-score value above 0.5 or 50%. In Figure 4, it can be seen that the fine-tuned models indobert-lite-squad and Roberta-1.5gb-tydiqa are two fine-tuned models with the highest accuracy and F-score values. In contrast, the fine-tuned models of Roberta-3gb-tydiqa, Roberta-1.5gb-squad, and Roberta-3gb-squad have average accuracy values below 50%. This proves that in the tests carried out, the three fine-tuned models that have average accuracy below 50% in several cases are not accurate in providing answer predictions.

The fine-tuned Roberta-1.5gb-tydiqa, which has an average accuracy value of 80% and an average F-score of 87%, has weaknesses, which sometimes can give wrong answers or no answers be given. However, this fine-tuned model has advantages for cases such as answers that are not available on the results of the information obtained – expected to be compared to the fine-tuned indobert-lite-squad model. The fine-tuned indobert-lite-squad model, this fine-tuned model has advantages when compared to the fine-tuned Roberta-1.5gb-squad model, where this fine-tuned model rarely gives the user wrong answers. However, the weakness is that the fine-tuned model still provides answers when the answers to user questions are not found in the information obtained.

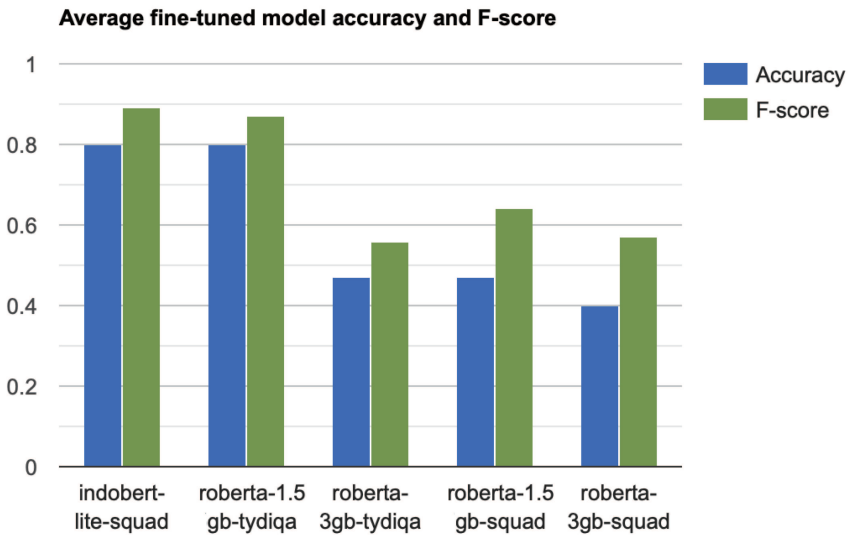

FIGURE 4. The average accuracy and F-score results of the fine-tuned models

The results found in this study show that the highest performing model is the fine-tuned indobert-lite-squad among the rest. This model yields an average peak accuracy of 80% and an F-score of 89%. The accuracy achieved in this study is higher compared to the previous study, with an average peak accuracy of 76.2% in [25]. The fine-tuned RoBERTa model of this study delivers an average F-score of 87%, which differs by −2.8% compared to the previous study using the SQuAD 2.0 dataset in [8]. However, that work is done using the English dataset.

5. **Conclusions.** This study successfully demonstrates the development of the Indonesian language Question Answering (QA) module integrated with Jacob's voice chatbot application. This is achieved by combining and expanding previous works on language processing, text mining, and question answering for the Indonesian language. The fine-tuned models that are applied to the Jacob are indobert-lite-squad and Roberta-1.5gb-tydiqa. The two models are chosen for Jacob based on the average accuracy and F-score values obtained from the testing and evaluation results.

1) Pre-trained RoBERTa model with Wikipedia dataset sizes 500MB and 363MB is not continued to the fine-tuning process due to the low accuracy of 0.041 and 0.032, respectively.
2) Pre-trained model cahya/Roberta-base-indonesian-1.5G is the best pre-trained model for fine-tuning the Indonesian SQuAD dataset with an average exact match value of 0.54 and an F-score of 0.59.
3) Pre-trained model cahya/Roberta-base-indonesian-1.5G is the best pre-trained model for fine-tuning the TyDi QA dataset with an average correct value of 154, the incorrect value of 97, and similar of 313.
4) Fine-tuned model Roberta-1.5gb-tydiqa is the best solution compared with other fine-tuned RoBERTa models in predicting answers with an accuracy value of 0.8 and an F-score of 0.87.
5) Fine-tuned indobert-lite-squad model is the best solution of all fine-tuned models used in predicting answers, with an accuracy value of 0.8 and an F-score of 0.89.

This study comes with a shortage of limited computer resources (CPU and RAM) for the computation of the models. Based on this shortage, the suggested future directions are as the following.

1) Exploration with the RoBERTa hyperparameters and training data with dataset size exceeding 10GB. This is to maximize the performance of the model.
2) Research to obtain a fine-tuned model that can perform question answering in Indonesian language using larger models, i.e., IndoBERT large and RoBERTa large.

Acknowledgment. The authors would like to thank Universitas Multimedia Nusantara for the support of this research work.

REFERENCES

[1] M. Dahiya, A tool of conversation: Chatbot, *International Journal of Computer Sciences and Engineering*, vol.5, no.5, pp.158-161, 2017.
[2] S. Mowla, I. Bedi and N. P. Shetty, A study on web mining tools and techniques, *J. Eng. Appl. Sci.*, no.Special Issue 2, pp.6135-6142, DOI: 10.36478/jeasci.2017.6135.6142, 2017.
[3] S. Dang and P. H. Ahmad, Text mining: Techniques and its application, *Int. J. Eng. Technol. Innov. (IJETI)*, vol.1, no.3, pp.22-25, 2014.
[4] 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.
[5] G. Y. N. N. Adi, M. Harley, V. Ong, D. Suhartono and E. W. Andangsari, Automatic personality recognition in Bahasa Indonesia: A semi-supervised approach, *ICIC Express Letters*, vol.13, no.9, pp.797-805, 2019.
[6] A. Chowanda, D. Suhartono, E. W. Andangsari and K. Z. bin Zamli, Machine learning algorithms exploration for predicting personality from text, *ICIC Express Letters*, vol.16, no.2, pp.117-125, 2022.
[7] B. Wilie et al., IndoNLU: Benchmark and resources for evaluating Indonesian natural language understanding, *arXiv.org*, arXiv: 2009.05387, 2020.
[8] Y. Liu et al., RoBERTa: A robustly optimized BERT pretraining approach, *arXiv.org*, arXiv: 1907.11692, 2019.
[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. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov and Q. V. Le, XLNet: Generalized autoregressive pretraining for language understanding, *arXiv.org*, arXiv: 1906.08237, 2019.
[11] M. Canonico and L. De Russis, A comparison and critique of natural language understanding tools, *Cloud Comput.*, no.4, pp.110-115, 2018.
[12] Octavany and A. Wicaksana, Cleveree: An artificially intelligent web service for Jacob voice chatbot, *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, DOI: 10.12928/TELKOMNIKA.v18i3.14791, 2020.
[13] A. Archilles and A. Wicaksana, Vision: A web service for face recognition using convolutional network, *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, DOI: 10.12928/TELKOMNIKA.v18i3.14790, 2020.
[14] F. P. Lovely and A. Wicaksana, Rule-based lip-syncing algorithm for virtual character in voice chatbot, *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, DOI: 10.12928/TELKOMNIKA.v19i5.19824, 2021.
[15] M. Hendronoto and A. Wicaksana, Implementation of ALBERT for text mining on Jacob voice chatbot, *ICIC Express Letters*, vol.15, no.10, pp.1029-1036, 2021.
[16] 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, 2019.
[17] S. R. Bowman, G. Angeli, C. Potts and C. D. Manning, A large annotated corpus for learning natural language inference, *Proc. of Conf. Empir. Methods Nat. Lang. Process. (EMNLP2015)*, pp.632-642, DOI: 10.18653/v1/d15-1075, 2015.
[18] M. Ott, S. Edunov, D. Grangier and M. Auli, Scaling neural machine translation, *arXiv.org*, arXiv: 1806.00187, 2018.
[19] R. Sennrich, B. Haddow and A. Birch, Neural machine translation of rare words with Subword Units, *arXiv.org*, arXiv: 1508.07909, 2016.
[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, *Language Models Are Unsupervised Multitask Learners*, Tech. Report, OpenAI, 2019.

[21] 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*, DOI: 10.18653/v1/d16-1264, 2016.
[22] P. Rajpurkar, R. Jia and P. Liang, Know what you don't know: Unanswerable questions for SQuAD, *The 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (ACL2018)*, vol.2, pp.784-789, DOI: 10.18653/v1/p18-2124, 2018.
[23] F. J. Muis and A. Purwarianti, Sequence-to-sequence learning for Indonesian automatic question generator, *arXiv.org*, arXiv: 2009.13889, 2020.
[24] J. H. Clark, M. Collins, D. Garrette and T. Kwiatkowski, TyDi QA: A benchmark for information-seeking question answering in Ty pologically Di verse languages, *Transactions of the Association for Computational Linguistics*, no.8, pp.454-470, 2020.
[25] R. Mahendra, A. F. Aji, S. Louvan, F. Rahman and C. Vania, IndoNLI: A natural language inference dataset for Indonesian, *arXiv.org*, arXiv: 2110.14566, 2021.

Author Biography

Benny Richardson received the B.Sc. degree in Informatics from the Department of Informatics at Universitas Multimedia Nusantara. A new researcher enthused in deep learning, reinforcement learning, computer security, and software engineering. He recently worked on software engineering for web and mobile applications.

Arya Wicaksana is a lecturer at the Department of Informatics at UMN. He received Master Degree in VLSI Engineering from Universiti Tunku Abdul Rahman. He successfully demonstrated the UTAR first-time success ASIC design methodology on a multi-processor system-on-chip project using 0.18 μm processing technology in 2015. His main research interests are blockchain applications and computational intelligence. He recently worked on a decentralized autonomous social media. He is affiliated with ACM and IEEE as a professional member. He has served as an invited reviewer in IEEE ACCESS, IJNMT, and IFERP and an invited author in IntechOpen and other scientific publications.