Performance Analysis and Development of QnA Chatbot Model Using LSTM in Answering Questions

M. Ilyas Tri Khaqi2, Nisa Hanum Harani2, Cahyo Prianto3
trikhaqi@gmail.com, nisa@ulbi.ac.id, cahyo@ulbi.ac.id
Universitas Logistik dan Bisnis Internasional

Submitted: 22 Jun 2023
Reviewed: 29 Jun 2023
Accepted: 30 Jun 2023

Abstract
This research aims to evaluate the performance of a Long Short-Term Memory (LSTM) based chatbot in answering questions (QnA). LSTM is a type of Recurrent Neural Network (RNN) architecture specifically designed to overcome vanishing gradient problems and can store long-term information. The method used is 5-fold cross-validation to train the chatbot model with 15 epochs at each fold using the dataset provided. The results showed variations in model performance at each fold. At the 5th fold, there was a decrease in performance with 84.63% accuracy, 96.36% precision, 64.9% recall, and 69.84% loss value. This finding shows that there is variability in the performance of the QnA chatbot model at each fold. In conclusion, the LSTM chatbot model can provide good answers with high accuracy and precision. Still, performance variations need to be considered in the use of this chatbot.

Keywords
QnA chatbot, LSTM, model performance, 5-fold cross-validation, performance variation.
A. Introduction

The use of chatbots has become an increasingly popular trend in various sectors, including customer service, education, and e-commerce [1][2][3]. Chatbots can provide efficient solutions for users who require information or assistance [4]. To improve chatbot capabilities, the use of the LSTM algorithm in the development of the QNA chatbot is the main focus.

The LSTM algorithm is one of the methods in the recurrence neural network that can consider the context of the conversation. With long-term memory capabilities, LSTM can recognize patterns and relationships between words in a sentence [5]. In addition, LSTM allows the chatbot to understand the user’s question as a whole, not just word by word, thus providing relevant and accurate answers.

Data is essential in developing an effective chatbot [6][7]. This study’s data source was the QnA fund and Indonesian Slang QnA data containing pairs of questions and answers created and collected from various sources. By utilizing this data, the LSTM chatbot can learn from different types of questions and contexts to provide comprehensive answers.

In addition to QnA data, Indonesian Slang data is also utilized in developing this chatbot. Slang is often used in daily communication, especially in social media and text messaging [8]. The chatbot can use this slang dictionary to recognize slang words in user questions and replace them with the right word. The dictionary helps the chatbot understand the question better and provide appropriate answers [9].

In a connected world with constantly evolving information, the need for chatbots that provide accurate and fast responses is increasing [10][11]. Accordingly, using an LSTM-based chatbot in QNA can improve the user experience in finding information or solving problems. With the ability to understand the context and adapt to available data, chatbots can provide relevant and efficient solutions in providing services [12].

The application of the LSTM chatbot in QNA assistant has vast potential in various sectors [13]. In the customer service industry, chatbots can help reduce response time and increase customer satisfaction by providing accurate and timely answers [3]. In education, chatbots can be learning assistants that help students get answers to their questions more efficiently [1][2]. In addition, chatbots can also be used in e-commerce platforms to provide product recommendations and better answer user questions [13].

QnA is a chatbot function designed to receive user questions and provide appropriate answers [14]. Using the LSTM algorithm to develop the QnA chatbot, the chatbot can process and analyze questions holistically, recognize patterns and relationships between words, and generate appropriate answers to allow users to get accurate and informative responses from the chatbot. In addition, Indonesian Slang is used to help the chatbot recognize and understand slang words frequently used in daily conversations.
B. Research Method

This research used the Microsoft Team Data Science Process method since it effectively manages and executes data science projects [15][16]. Besides, this method is also designed to complete research in a structured and organized manner, from understanding the business to implementing solutions. The following figure is the flow or stages of research using the method.

![Research Flow Diagram](image)

**Figure 1. Research flow**

The first stage understands the business problem and project objectives in this method. Then identify questions that must be answered, such as how the chatbot can improve the user experience and the expected response of the chatbot, and clarify relevant business requirements for further. The second stage is to collect and analyze the data used in the LSTM chatbot code, explore the available data, understand its structure and quality, and look for patterns or trends supporting problem-solving. This research uses 'data.csv' and 'slangindonesia.csv' for data sources. The source of 'data.csv' is based on daily questions, jokes, and understanding simple things. While 'slangindonesia.csv' data was taken from Kaggle [https://www.kaggle.com/datasets/sodolanangbjkatio/slang-indonesia](https://www.kaggle.com/datasets/sodolanangbjkatio/slang-indonesia), however, in the 'slangindonesia.csv' data, there are some structural changes and data additions made. The third stage consists of a series of steps. They are to prepare raw data for analysis or modelling [17]. Such include cleaning data from noise, integrating data from various sources, and performing data processing such as removing and replacing missing values and data transformation. The main purpose of this stage is to ensure that the data used is clean, structured, and ready for use. The fourth stage selects the appropriate model or algorithm to solve existing business problems. One is the LSTM model on the chatbot—implementing LSTM chatbot code that includes LSTM. This model allows the chatbot to learn the patterns and context of conversations to provide users with more relevant and interactive responses. The LSTM is a type of RNN architecture specifically designed to overcome the "vanishing gradient" problem that often occurs in ordinary RNNs [18]. LSTM can recognize complex patterns and retain long-term information in data sequences [19]. The formula used in LSTM can be explained as follows:

a) Forget Gate

\[ f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \]  \hspace{1cm} (1)
Forget gate sets the extent to which previous information will be forgotten. \( f_t \) is a value between 0 and 1 that determines how much information will be removed from the previous LSTM cell [19].

b) Input Gate

\[
\begin{align*}
  i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
  \hat{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)
\end{align*}
\]

The input gate allows the model to decide to what extent new information will be added to the LSTM cells. \( i_t \) is a value between 0 and 1 that determines how much new information will be integrated, while \( \hat{C}_t \) is a candidate vector of values that can be added [19][20].

c) Cell Information

\[
C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t
\]

Cell Information is new information proposed to be incorporated into the cell state at each iteration. This allows the LSTM model to adjust relevant information and maintain long-term memory by ignoring less relevant information [19].

d) Output Gate

\[
\begin{align*}
  o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
  h_t &= o_t \cdot \tanh(C_t)
\end{align*}
\]

The output gate regulates the extent to which the information in the LSTM cell will be output. \( o_t \) is a value between 0 and 1 that determines how much information will be expressed through the output, and \( h_t \) is the final result which is a representation of the information that will be sent to the next layer or as the final output of the LSTM [19][20].

![Figure 2. LSTM Architecture](image-url)
In the formula, $W$ and $b$ are the weights and biases that must be learnt during the LSTM training process. $\sigma$ is the sigmoid activation function, while tanh is the hyperbolic tangent activation function used to scale the values.

The last stage is the deployment process, which involves implementing the LSTM chatbot into the production environment. Integrating the chatbot into existing systems, ensuring its optimized performance, and providing documentation and guidance on using the chatbot to stakeholders [21].

C. Result and Discussion
1. Business Understanding
   In the context of LSTM chatbots, business understanding focuses on identifying whether the answers the chatbot must provide are appropriate. The questions to be answered may include things like how the chatbot can improve user experience, how the chatbot can provide quick solutions, and how the chatbot can improve service efficiency. Examples of such questions include questions with simple things like greetings, joking words, and questions regarding multiple context questions from understanding.

2. Data Acquisition and Understanding
   Data Understanding Process data.csv and slangindonesia.csv The Data Preparation process in developing LSTM chatbot using data.csv and slangindonesia.csv involves several important steps. The following is an explanation of the data preparation process from the data.csv and slangindonesia.csv files:

   a) QnA Data
   The first step is to read the "data.csv" file using the appropriate method for reading CSV files. Then, the data must be separated into columns: "question" and "answer". Next, clean the data of unnecessary characters, such as spaces or irrelevant punctuation.

   ![Figure 3. QnA Data](image)

   b) Indonesian Slang Data
   The data preparation process for the file "slangindonesia.csv" also starts with reading the file using the appropriate method. After that, the data needs to be
separated into two columns, namely "slang" (slang words) and "non-slang" (non-slang words). If there are special characters or unnecessary punctuation marks, perform text normalization or replacement of slang words and non-slang equivalents for consistency in the data processing.

![Figure 4. Indonesian Slang Data](image)

Then the process of collecting and analyzing data to understand the characteristics, structure, and quality of existing data, The process of understanding data involves several steps:

a) CSV File Reading

The programme code uses the 'csv' reading module to read the CSV file containing the questions and answers. In the first line, the programme code opens the file 'data.csv' and uses 'csv.reader' to read the file's contents with a '|' (pipe) separator.

b) Formation of Question and Answer Dictionary

Each line in the read CSV file is broken down into questions and answers. The question mark at the end of the question is removed, and then the processed questions and answers are entered into the 'qna_dict' dictionary, with questions as keys (lowercase) and answers as values.

c) Slang and Non-Slang CSV File Reading

The next process is still the same; the programme code uses the 'csv' module to read the CSV file containing Slang and non-slang words. On the next line, the code opens the file 'slangindonesia.csv' and uses 'csv.reader' to read the file's contents with a ',' (comma) separator. And the lines are read one by one.

d) Formation of Slang and Non-Slang Dictionary

Each line in the CSV file that has been read is broken down into Slang and non-slang words. Then the processed ones are entered into the 'slang_dict' dictionary, with the slang word as the key (lowercase) and the non-slang word as the value.

e) Text Preprocessing

In the programme code, this step is defined in the function 'preprocess_text(text)' to perform text processing. This function accepts text as input and performs several preprocessing steps, including:
1. Change all letters to lowercase.
2. Remove punctuation and special characters using a regular expression (`re.sub()`) with the pattern `r'[^a-zA-Z0-9\s]'`.
3. Perform word tokenization using the `word_tokenize()` function of the NLTK module. The text will be broken down into a collection of words.
4. Removing stopword words. Stopwords are words that do not provide important meaning in text processing. The list of Indonesian stopwords is downloaded using the function `set(stopword.words('indonesian'))`. So words in the text are retrieved only if they are not included in the stopwords. This is done using list comprehension and the `if word not in stopwords` condition.
5. Perform lemmatization using `WordNetLemmatizer` from NLTK (`WordNetLemmatizer()`). It aims to convert words into their base form (lemmatization).
6. Stemming using `PorterStemmer` of NLTK (`PorterStemmer()`) aims to convert words into root words (stem).
7. Recombining the processed words into strings using the `join()` function. Words will be joined with spaces as separators between words.

f) Slang Replacement
In the process of replacing Slang to identify Slang or abbreviations with their true meaning, including:
1. Break the text into tokens using the `split()` function. Each word in the text will be a separate token.
2. Repeat (`loop`) for each token in the text.
3. Tokens are stored in the variable `token`.
4. Check if the token is included in the slang dictionary (`slang_dict`).
5. If a slang token is found in the slang dictionary, it is replaced with the corresponding non-slang token in the dictionary. The replacement is done by changing the token value at the i-th index in `tokens`.
6. The slang replacement is complete; the modified tokens are merged back into text using the `join()` function with space as a separator between the tokens.
7. The modified text is returned as the output of the function.

3. Modeling
Chatbot LSTM involves building an LSTM model that will be used to solve problems in the chatbot context. The modelling process involves several stages as follows:

a) Hyperparameter Initialisation
Initialize the hyperparameters for the model, such as the number of neurons, dropout rate, learning rate, number of epochs, and batch size [22].

b) Training Data Formation
Convert question and answer data from the dictionary into training data that will be used to train the model. The question data becomes the input, while the answer data becomes the corresponding output [23].
c) Cross-Validation
Perform cross-validation on the training data. The training data will be divided into several folds, and the model will be trained and evaluated using cross-validation techniques to obtain more accurate performance estimates [24].

d) Tokenization
Transforming text into a sequence of tokens (words) used Tokenizer from Keras to tokenize the question data [22].

e) Formation of Question and Answer Index Dictionary
Transforming question and answer data into an indexed dictionary using Tokenizer. Each word in the question and answer data is given a unique index to form a dictionary used in the training process.

f) Padding
The question and answer data uses the pad_sequences() function from Keras. Padding is used to equalize the length of the token sequence in each data so that it can be processed by the model with a consistent size [22].

g) Model Building
Build the LSTM model using Sequential from Keras. The model consists of an Embedding layer, LSTM, and two Dense layers. The Embedding layer is used to convert the index dictionary into vectors. The LSTM layer is used to process the token sequence, and the Dense layer is used to generate the output [25].

```
Model: "sequential_23"

Layer (type)             Output Shape    Param #     
=================================================================
embedding_23 (Embedding)   (None, 10, 100)  343400
lstm_23 (LSTM)               (None, 128)    117248
dense_46 (Dense)            (None, 64)      8256
dense_47 (Dense)            (None, 8902)    258630
=================================================================
Total params: 722,534
Trainable params: 722,534
Non-trainable params: 0
```

**Figure 5.** Model building results

h) Model Compilation
Compile the model using the compile() function. The model is compiled with the Adam optimizer, categorical_crossentropy loss function, and loss, accuracy, precision, and recall metrics.

i) Model Training
Train the model using the fit() function. The processed training data is fed into the model to be trained. Training is done in several epochs using a
predefined batch size. During model training, the fit() function contains the parameters 'padded_questions', 'padded_answers', 'epoch=num_epochs', and 'batch_size=batch_size'.

j) Model Evaluation
After the training, the model is evaluated using the previously separated testing data. The evaluation is done by calculating the model's accuracy, precision, and recall against the testing data. Through this Evaluation process, the quality and performance of the LSTM chatbot can be comprehensively evaluated. This evaluation ensures that the chatbot can provide appropriate, relevant, and responsive responses to users. If the evaluation results do not meet the set standards, a refinement and iteration stage can be completed to improve the chatbot's performance before further implementation. The process involves several stages, as follows:

1. Taking Values Before Using Cross-Validation
The following is a breakdown of the final values of the chatbot training results before using the cross-validation process. In Table 1, it can be seen that trials 1 to 5 show signs of overfitting, as there is a significant difference between high training accuracy and lower testing accuracy [26]. Overfitting occurs when the model is too "familiar" with the training data and cannot generalize well to new data.

<table>
<thead>
<tr>
<th>Table 1. Without Using Cross-Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

2. Retrieval of Values After Using Cross-Validation
After the model training is complete, the next process takes the loss, accuracy, precision, and recall values from the 'history' object, which contains the model's history. The process uses Cross Validation to evaluate the model objectively and provide different results from previous experiments, as seen in Table 2. Based on the final results of 5 folds, these values are used to evaluate model performance.

<table>
<thead>
<tr>
<th>Table 2. Using Cross-Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fold</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
k) Model Saving
   Once the training is complete, the LSTM method is saved in '.h5' format using the 'save' method.

l) Model Integration with Telegram
   1. Import the required libraries and modules.
   2. Initialize the Telegram bot object using the bot token.
   3. Loading the LSTM model from the .h5 file.
   4. Reading the slang dictionary from the CSV file.
   5. Reads questions and answers from the training data dictionary from the CSV file.
   6. Build a tokenizer to process the question text.
   7. Build a dictionary to map question and answer indexes.
   8. Define the text pre-processing function.
   9. Defined the handler for regular messages of the LSTM model.
   10. Run the Telegram bot using the polling method.

4. Deployment
   The deployment process in chatbot development is the stage where the chatbot model that has been trained and evaluated is deployed for use by users or customers [21]. This process involves preparing a suitable production environment, such as a server or hosting platform, and integrating relevant communication systems or channels. The main goal of the deployment process is to ensure that the chatbot runs well, provides a good user experience, and provides relevant and accurate solutions in interaction with users on an ongoing basis.

![Image of Telegram chat]

Figure 6. Results of Model Integration with Telegram
D. Conclusion

The results of the LSTM chatbot with 5-fold cross-validation show the model's performance in each fold and epoch performed. In fold 1 (one), the model accuracy reached 93.56% with a precision value of 97.37% and a recall of 85.67% after 15 epochs. In fold 2 (two), the accuracy increased to 94.55%, with a precision of 97.29% and a recall of 86.85%. Fold 3 (three) also performed well, with 93.88% accuracy, 97.63% precision, and 85.93% recall. In fold 4 (four), the accuracy increased to 96.19% with 98.07% precision and 91.01% recall. However, at fold 5 (five), there is a decrease in performance with 84.63% accuracy, 96.36% precision, and 64.9% recall. These results show the variation of the model performance in each fold and can be used to identify the consistency or variability in the model training results. In addition, it should be noted that accuracy, precision, and recall are commonly used evaluation metrics in machine learning. A more comprehensive understanding of how the chatbot LSTM model behaves and is consistent in predicting the data can be obtained by looking at the performance results at each fold and epoch.

E. References


[8] R. Jiamthaphaksin, P. Setthawong, and N. Ratanasawetwad, "A system for popular Thai slang extraction from social media content with n-gram based tokenization," in 2016 8th International Conference on Knowledge and Smart


[20] A. W. Sugiyarto and A. M. Abadi, "Prediction of Indonesian palm oil production...


