Chatbot Application "Konco Ngobrol" in Ngapak Javanese Using LSTM Model Approach

Salman Baehaqi[1], Hindayati Mustafidah[2]*, Maulida Ayu Fitriani[3], Agung Purwo Wicaksono[4]

[1,2,3,4]Department of Informatics Engineering, Faculty of Engineering and Science, Muhammadiyah University of Purwokerto, Banyumas, Central Java, Indonesia
E-mail: [1]salmanbaehaqi1502@gmail.com, [2*]h.mustafidah@ump.ac.id, [3]maulidaayuf@gmail.com, [4]wicaksono@ump.ac.id

(Received: 15 May 2025, revised: 11 Jun 2025, accepted: 12 Jun 2025)

Abstract

Regional languages are an important part of cultural identity, increasingly marginalized in the digital era. Ngapak Javanese, a local dialect, is declining in use in Banyumas, Tegal, Purbalingga, and nearby areas. This research aims to develop an interactive text-based chatbot named Konco Ngobrol that uses Ngapak Javanese as the main conversational language. This chatbot is built using the Long Short-Term Memory (LSTM) model approach, which excels at understanding conversation context in the form of sequential data. The dataset is composed of many sentence patterns with a mixed language style of Indonesian and Javanese Ngapak. The research stages include data collection, pre-processing (tokenization, padding, and label encoding), building and training the LSTM model, model evaluation, prediction using a combination of LSTM and fuzzy matching, and deployment into a Flask-based application with an interactive chat interface. The evaluation results show that the LSTM model can achieve an accuracy of up to 99.04% with a low loss value, and can provide contextually appropriate responses. The fallback system also successfully handled unknown inputs well through fuzzy logic. The Konco Ngobrol chatbot not only serves as a medium for entertainment and communication but also as a tool for preserving digital culture by reintroducing local dialects into the modern technology ecosystem.

Keywords: Chatbot, Flask, Fuzzy Matching, LSTM, Ngapak Javanese.

I. INTRODUCTION

Everyday conversation is a fundamental form of communication used by humans to exchange information, express emotions, and build social relationships in daily life. In the field of linguistics, everyday conversations are characterized by their spontaneity, informality, and contextual nature, shaped by the social and cultural environment in which they occur [1]. The language used in such conversations often reflects local cultural identity, including the use of specific dialects or language varieties such as Ngapak Javanese, which is distinctively spoken in the regions of Tegal, Purbalingga, Banyumas, and surrounding areas. Conversations not only serve a communicative function but also play a role in preserving local values embedded within the structure of the language.

Etymologically, the word "*percakapan*" (conversation) derives from the root word "*cakap*", which in Indonesian means to speak or to converse. The addition of the prefix "*per*" and the suffix "*-an*" forms a noun that denotes an activity or process of two-way communication. People use everyday conversations not only to convey verbal messages but also to express cultural nuances that vary by region [2]. For instance, in Ngapak Javanese, the phrase "*arep lunga neng endi, cah?*" translates to "where are you going, bro?" in Indonesian. The words "*arep*" (want), "*lunga*" (go), and "*neng endi*" (to where) highlight the distinctive Ngapak sentence structure, which is known for its blaka suta (straightforward) nature and minimal speech level stratification. This expression reflects the egalitarian and open communication style among Ngapak speakers. Preserving such linguistic forms through technologies like chatbots plays a vital role in maintaining their existence, especially as younger generations are more accustomed to digital communication [3].

Local languages are cultural heritages that carry not only linguistic value but also the social identity and historical legacy of a community. One of the major challenges in the current era of globalization is the declining use of regional languages in daily life, particularly among younger generations [4]. Although Javanese still has a large number of speakers, it faces preservation challenges, especially in dialectal variants such as Ngapak Javanese [5]. To address this issue, the use of

technology, particularly artificial intelligence (AI), offers an innovative solution to reconnect regional languages with society, especially through interactive platforms such as chatbots [6].

A chatbot is an AI-based system designed to interact with users through text or voice in a conversational format that mimics human communication. Advancements in the field of Natural Language Processing (NLP) have enabled chatbots to understand user intent, process conversational context, and generate relevant and contextually appropriate responses [7]. In the context of regional languages, chatbots serve not only as communication tools but also as educational platforms and instruments for cultural preservation [8].

The development of artificial intelligence (AI) offers new opportunities to overcome communication barriers. AI has advanced the ability to interact with humans through conversations that resemble natural communication. This process relies on methods from Natural Language Processing (NLP). Several studies, such as those outlined in [9], demonstrate the effectiveness of NLP in processing textual data. One of the key NLP applications enabling more natural interaction between users and systems is the chatbot, which can understand user intent and provide relevant responses.

One of the most relevant and effective models for implementing NLP in chatbots is the Long Short-Term Memory (LSTM) model. LSTM is a variant of Recurrent Neural Networks (RNN) that excels at handling long-term dependencies in sequential data such as language [10]. Numerous studies have shown that LSTM is one of the most effective deep learning approaches for building chatbot systems, due to its ability to understand sentence context and manage sequential data like human conversations [11]. LSTM utilizes a specialized memory structure known as gates (input, forget, and output) to regulate relevant information throughout the conversational process [12].

The development of chatbot systems based on local languages such as Ngapak Javanese faces significant challenges due to the inconsistencies in spelling and pronunciation among speakers, including variations in vocabulary, intonation, and sentence structure. These conditions make it more complex for the system to understand user intent, requiring specialized approaches in system design so that the chatbot can adapt to the linguistic diversity present. One highly relevant approach to addressing these challenges is fuzzy matching. Fuzzy matching is a string-matching technique that enables the system to recognize words or phrases that are similar, even if they are not spelled exactly the same [13]. This technique is particularly useful in chatbot systems to improve the accuracy of matching user inputs with predefined patterns in the dataset, especially in regional languages like Ngapak, which contain numerous phonetic and speech variations. For example, a user might write "*arep dolan*," "*arep dolen*," or even "*arep dolan nang*," and through fuzzy matching, the system can still identify the same intent: "want to go out and play." This is crucial for supporting flexible, natural conversations in chatbots designed for non-standardized languages.

This study supports previous findings that demonstrate the effectiveness of LSTM in the development of multilingual chatbots with high accuracy. For example, [14] notes that LSTM can achieve accuracy rates of up to 100% in classifying intents for an Indonesian-language chatbot. Additionally, [15] developed a Named Entity Recognition (NER) system based on Bi-LSTM that achieved an F1 score of 87.44%, reinforcing the argument that this model has high capabilities in natural language processing. A study by [16] also proved that an LSTM-based chatbot for academic information achieved a validation accuracy of up to 99% using a user validation method, demonstrating that this approach is well-suited for complex, contextual, text-based conversations on a local scale.

A study by [17] on Javanese-language chatbots with polite speech levels (krama alus) reveals that the development of chatbot systems focused on local culture can play a significant role in enhancing the understanding and use of regional languages in the digital realm. Thus, chatbot technology can serve as an effective means to promote the preservation of regional languages while ensuring their continued use in the future. However, to date, there has been very little research specifically focused on developing Ngapak Javanese-based chatbots with an interactive approach and casual, conversational style typical of peer groups. This highlights a research gap that needs to be addressed.

Although numerous studies have been conducted on the development of chatbots based on regional languages, most of them are still limited to the use of more formal language or dialects with standardized writing systems. Research that specifically addresses the Ngapak dialect, characterized by its informal tone, blunt expression (*blaka suta*), and minimal speech level differentiation, is still extremely scarce. This indicates a pressing need to design AI-based communication systems capable of capturing the linguistic complexity of Ngapak Javanese while accommodating the sociocultural dynamics of its speakers. Therefore, the development of a chatbot using Long Short-Term Memory (LSTM) technology, combined with a fuzzy logic approach, offers a promising solution to the challenges of understanding non-standard conversational patterns and facilitates the preservation of local culture through an interactive and contextually relevant digital medium.

This study aims to develop a conversational chatbot capable of interacting in Ngapak Javanese in a contextual and natural manner by utilizing a Long Short-Term Memory (LSTM) model to understand word sequences and conversational context. Beyond serving as a communication tool, the chatbot is designed to function as an educational and cultural preservation medium powered by technology. The research contributes significantly to the field of Natural Language Processing (NLP) for regional languages by introducing an integrative approach that combines LSTM and fuzzy matching to handle non-standardized linguistic input. Furthermore, the system presents a chatbot architecture that can be adapted to other local dialects, making it a promising model for developing chatbots based on local wisdom across Indonesia.

DOI: 10.34148/teknika.v14i2.1258

Based on the identified problems, this research offers a novel approach by focusing on the Ngapak dialect as the primary conversational language in the chatbot system, an area that has been largely overlooked in previous studies. In addition to constructing a dataset that combines Indonesian and Ngapak Javanese with the distinctive speech style of young speakers, this study implements a dual-path processing strategy using both fuzzy logic and LSTM. This approach allows the system to handle inputs with high variability in spelling and language style challenges that are commonly encountered in the processing of regional languages lacking standardized orthography. The combination of machine learning techniques and string similarity-based heuristics provides a robust solution for managing non-standard natural language variations, making this research an original contribution both in terms of linguistic content and technological approach.

II. METHODOLOGY

This study was conducted through several stages to develop a Ngapak Javanese-language chatbot application called "Konco Ngobrol" using the LSTM (Long Short-Term Memory) model approach. The chatbot development method follows the AI Project Cycle, which consists of several stages: data collection, pre-processing, modeling, evaluation, and deployment [11]. The AI Project Cycle process is illustrated in Figure 1.

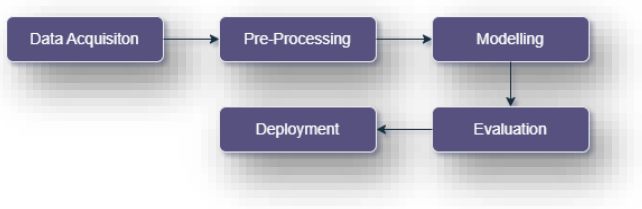


Figure 1. Stages of the AI Project Cycle

A. Data Acquisition

The data used in this study was collected and consists of three main elements: tags as conversation labels/topics, patterns as inputs that may be provided by users, and responses as the replies given by the chatbot. The conversation data in this study was organized with an approach that prioritizes the characteristics of informal conversations typical of communities in the Banyumas, Tegal, and surrounding areas. The choice of vocabulary and sentences such as "*arep dolan neng endi,*" "*yo wis lah bro,*" or "*ngapain mikir rumit, sing santai wae*" reflects linguistic traits discussed in studies by [3] and [4], which emphasize that the Ngapak dialect has an expressive communication style that tends to be straightforward.

This dataset consists of 79 tags (intents) representing a variety of intent, including *"sapaan_awal," "candaan," "nasihat,"* and fallback responses. Each tag contains between 20 to 30 sentence patterns that serve as representations of user inputs, along with several response options that the chatbot can generate. In total, the dataset includes more than 1,500 sentence patterns. The language used in the patterns is a 50:50 mix of casual Indonesian and Ngapak Javanese, designed to mimic the conversational style of young people in informal social settings. The dataset was manually compiled, with careful attention to vocabulary diversity, non-standard spelling variations, and the direct, expressive speech style typical of the Ngapak dialect. All data is formatted in JSON to be compatible with LSTM model training.

B. Pre-processing

Pre-processing the data to prepare it in a numerical format that can be processed by the LSTM model. This process begins with tokenization using the Tokenizer library from Keras, which converts each sentence in the patterns section into numerical tokens. Next, padding is applied using the pad_sequences function to ensure all inputs have the same length, making them compatible with the model's input structure. The labels for each intent (tag) are transformed into numerical representations using LabelEncoder. This stage results in two main components: the input data, which consists of tokenized sequences of numbers, and the output data, which consists of numerical labels, both ready to be used in the training process.

C. Modelling

The chatbot model is built using the LSTM (Long Short-Term Memory) architecture with a layered neural network approach. Long Short-Term Memory (LSTM) is a model specifically designed to address the vanishing gradient problem and support context understanding in data with temporal sequences [18]. This model is highly effective in processing information that arrives in sequential form, such as in text or conversation analysis. The choice of LSTM for this chatbot development is based on its ability to achieve high accuracy, optimal performance, and fast responses, which are essential for ensuring smooth interactions with users.

To improve accuracy and user experience, the process incorporates fuzzy matching using the fuzzywuzzy library, which enables the chatbot to match user inputs with patterns in the dataset. Fuzzy matching is a text-matching technique that allows the system to find similarities between two strings even if they are not exactly the same or closely related. In this study, the fuzzy matching technique is applied using the fuzzywuzzy library, which internally utilizes the Levenshtein Distance algorithm to calculate the similarity between strings [19]. The use of fuzzy matching in this study is grounded in a core challenge of natural language processing (NLP) involving local dialects, namely, the high degree of variation in spelling, sentence structure, and the use of non-standard terms. Javanese Ngapak, like many other regional dialects, lacks a standardized orthography, allowing speakers to express the same meaning in multiple written or spoken forms. This leads to a significant possibility of mismatch between user input and the predefined sentence patterns in the dataset, especially when users write phrases in ways that differ substantially from the model’s training data. For instance, the

sentence *"arep dolan neng endi?"* may be written as *"arep dolen neng ndi?"* or even *"arep dolan nang endi bro?"*, all of which convey the same semantic intent but differ in literal form. Such variations pose a challenge for LSTM-based models, which rely heavily on the sequential order of trained tokens and may struggle to recognize these alternative expressions directly.

To address this challenge, fuzzy matching was implemented as an initial layer in the intent prediction process to compare user input against the list of predefined patterns in the dataset. By utilizing the Levenshtein Distance algorithm through the fuzzywuzzy library, the system is able to calculate the similarity between strings and still recognize user intent despite minor variations in spelling. This approach enables the chatbot to generate more tolerant predictions with respect to linguistic variation and typographical errors, thereby improving the system's reliability in informal conversational contexts. Fuzzy matching also serves as an additional verification mechanism, particularly when the LSTM model exhibits low confidence in its predictions, allowing the system to still deliver relevant responses even when the input does not explicitly match the training patterns, as illustrated in Figure 2.

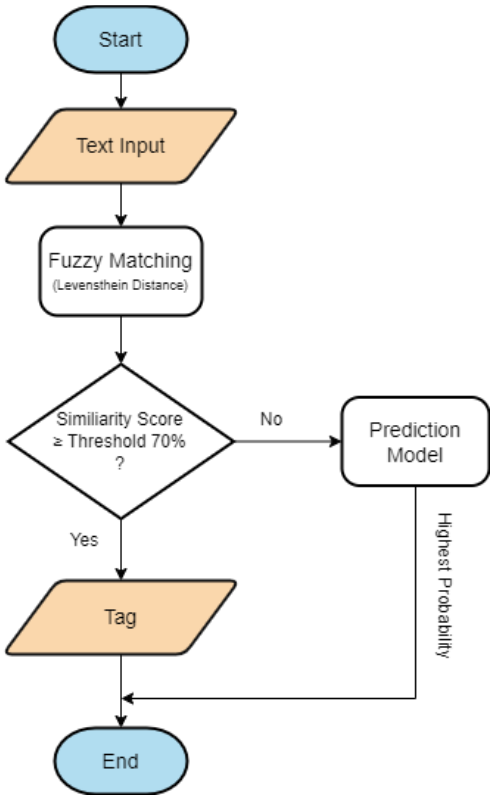


Figure 2. Stages of the Process From Input to Output

Figure 2 shows the process that combines fuzzy matching with an LSTM model prediction. At the initial stage, the user provides input in the form of text. The system then matches all patterns in the dataset using the fuzzy matching method. Fuzzy matching is a string matching technique that allows for minor differences between words. This technique is useful in handling typos, spelling variations, or non-standard word arrangements. In its implementation, fuzzy matching is used to match user questions or inputs with patterns found in the chatbot dataset. This is done by calculating the similarity level between strings, for example, using the Levenshtein Distance algorithm. Levenshtein distance measures the minimum number of operations (insertion, deletion, or substitution of characters) required to transform one string into another [20]. Suppose there are two strings, namely string A of size n and string B of size m. Then the Levenshtein distance $D(i,j)$ is defined as Formula 1 [20]:

$$D_{i,j}=\begin{cases}\max_{(i,j)} & \text{if } \min(i,j)=0\\ \min\begin{cases}D(i-1,j)+1\\ D(i,j-1)+1\\ D(i-1,j-1)+1_{(a_i\neq b_j)}\end{cases} & \text{otherwise.}\end{cases} \quad (1)$$

After the distance is calculated, the similarity score is normalized into a percentage using Formula 2 as follows:

$$Similiarity\ score = \left(1-\frac{Levenshtein\ Distance}{\max\left(Lenght(a),lenght(b)\right)}\right)\times 100\% \quad (2)$$

If the similarity score is ≥ 70%, the system directly uses the tag resulting from the fuzzy match as the predicted intent. In this context, a tag serves as a label used to group several input sentence patterns (patterns) and corresponding response sentence patterns (responses) that the chatbot provides when the user's input matches one of the patterns under a given tag. Each tag represents a single intent. For example, the tag *"sapaan_awal"* is used to indicate greetings such as *"halo," "hai,"* or *"assalamualaikum,"* while the tag *"end_conversation"* is used for farewell expressions such as *"dadah," "pamit yo,"* or "bye." However, if the fuzzy match score is below the threshold, the system proceeds with intent prediction using the LSTM model. In other words, when the similarity score is < 70%, the system delegates the prediction task to a pre-trained LSTM model by selecting the tag with the highest predicted similarity score. The output from both pathways (fuzzy matching and LSTM prediction) is then used to determine the chatbot's response. Additionally, fuzzy matching is employed as a supplementary validation method, particularly when the model's confidence score is below 40%. In such cases, fuzzy matching is not used as the primary basis for intent prediction; instead, the system defers to the LSTM model trained on intent data to determine the most appropriate response.

The chatbot's responses to most user inputs were generally accurate and relevant to the context of the queries. However, some inputs containing ambiguous phrasing or code-mixing occasionally resulted in low similarity predictions. To address this issue, a fallback system is implemented. If the predicted tag similarity score is below 40%, the system assumes that the input cannot be reliably recognized and responds with a neutral fallback tag. The LSTM model calculates the probability of each intent tag based on the input patterns it has learned during training.

DOI: 10.34148/teknika.v14i2.1258

D. Evaluation

The model training was conducted using a training dataset encompassing the entire learning process for up to 200 epochs. During training, several metrics were used to evaluate the performance of the LSTM model, including the "accuracy" metric, which measures the precision of the model's predictions, and the "sparse_categorical_crossentropy" loss function, which calculates prediction error. Additionally, the optimization process utilized the Adam optimizer to enhance the efficiency of model convergence. Accuracy and loss values during training were recorded and visualized using graphs to evaluate the model's performance.

E. Deployment

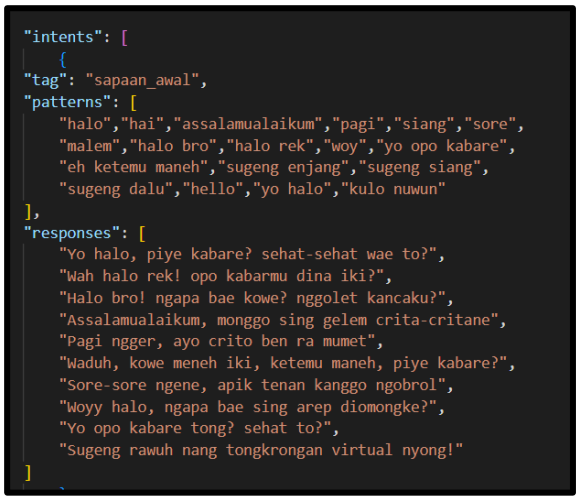
The chatbot implementation will be carried out using the Flask framework, as recommended in [21]. For the user interface, the application will be developed using web technologies such as HTML, CSS, and JavaScript, aiming to provide an interactive and responsive user experience. Backend development also requires an optimized development environment to ensure the chatbot functions efficiently. This environment includes various libraries and frameworks, such as Flask for web management, NumPy for numerical computations, TensorFlow for machine learning model processing, scikit-learn for machine learning techniques, fuzzywuzzy for fuzzy matching, python-Levenshtein for calculating Levenshtein distance, and matplotlib for data visualization.

III. RESULT AND DISCUSSION

A. Data Acquisition

The dataset used in the development of this chatbot was specifically designed to reflect the informal speech style of young people who use a mixture of Indonesian and Ngapak Javanese. Structured in JSON format, the dataset consists of 79 intent tags, each containing 20 to 30 sentence patterns and a set of contextually appropriate responses. Each component plays a distinct role in supporting the chatbot system. Intents indicate the purpose of the conversation, tags serve as category labels, patterns contain examples of user inputs, and responses determine the replies to be provided by the chatbot. The dataset includes over 1,500 sentence patterns in total, with more than 500 response variations. The complete structure of the dataset can be seen in Figure 3.

In Figure 3, there is a patterns component in the dataset that serves to capture various phrases or sentences that are likely to be used by users when interacting with the chatbot. Variations in conveying intent are important so that the chatbot can recognize the various ways people express messages. For example, in the intent *"sapaan awal,"* patterns can include phrases like *"Hai," "Halo,"* or *"Sugeng enjang"* that are commonly used as forms of initial greetings. Meanwhile, the responses component serves to provide relevant and responsive replies, tailored to the user's emotional state or situation. For example, the chatbot can respond with a sentence like *"Yo halo, piye kabare? Sehat-sehat wae to?"* or *"Wah halo rek! Opo kabarmu dina iki?"* These responses are constructed with careful consideration of the *blaka suta* speaking style and the distinctive vocabulary of the Ngapak dialect. With a clear and organized dataset structure, the AI model training process can run more effectively, giving the chatbot the ability to respond more flexibly and accurately in various conversational situations.



```
"intents": [
    {
"tag": "sapaan_awal",
"patterns": [
    "halo","hai","assalamualaikum","pagi","siang","sore",
    "malem","halo bro","halo rek","woy","yo opo kabare",
    "eh ketemu maneh","sugeng enjang","sugeng siang",
    "sugeng dalu","hello","yo halo","kulo nuwun"
],
"responses": [
    "Yo halo, piye kabare? sehat-sehat wae to?",
    "Wah halo rek! opo kabarmu dina iki?",
    "Halo bro! ngapa bae kowe? nggolet kancaku?",
    "Assalamualaikum, monggo sing gelem crita-critane",
    "Pagi ngger, ayo crito ben ra mumet",
    "Waduh, kowe meneh iki, ketemu maneh, piye kabare?",
    "Sore-sore ngene, apik tenan kanggo ngobrol",
    "Woyy halo, ngapa bae sing arep diomongke?",
    "Yo opo kabare tong? sehat to?",
    "Sugeng rawuh nang tongkrongan virtual nyong!"
]
```

Figure 3. Dataset in JSON Format

During the dataset development process, variations in sentence structure and vocabulary were carefully considered to enable the system to recognize non-standard inputs, both in terms of spelling and pronunciation. All data were manually compiled and validated through a reading test involving native speakers from the Tegal, Banyumas and Purbalingga regions to ensure the authenticity of the dialect and its cultural relevance. The data were also categorized based on conversation topics intent such as "*sapaan_awal*," "*makanan*," "*hobi*," "*candaan*," "*cuaca*," and fallback responses. This classification allows the chatbot to handle various informal conversation scenarios in a contextual and natural manner. With a well-structured and organized dataset, the AI model training process can be optimized, resulting in a chatbot system capable of delivering accurate and relevant responses across a wide range of everyday communication contexts.

B. Pre-processing

The first stage begins with the dataset loading process, which involves opening the dataset file while ensuring that non-standard characters, such as those found in regional languages, can be properly read. The file is loaded so that its contents can be accessed as a dictionary structure in Python. The dataset contains a list of intents, each of which includes a tag, a set of example sentences (patterns) serving as user input, and corresponding responses that the chatbot will provide during interactions.

The next step is data preprocessing, which involves separating all input sentences (patterns) into a list of sentences and storing their tags in a list of labels. In this way, each input sentence is prepared for further processing and already has the appropriate label or category. This data then needs to be

converted into numerical form because machine learning models cannot directly process raw text. Therefore, tokenization is performed, which is responsible for converting each word into a number based on the token dictionary. The results of the tokenization can be seen in Table 1.

Table 1. Tokenization Results

Sentence	Tokenization
'Yo apa kabare'	*'Yo'* : 1 , *'apa'* : 2 , *'kabare'* : 3
'Lagi apa'	*'Lagi'* : 4 , *'apa'* : 2
'Tak tinggal sek'	*'Tak'* : 5 , *'tinggal'* : 6 , *'sek'* : 7

After all sentences are converted into sequences of tokens, padding is applied using the pad_sequences function to ensure that all input sequences have the same length. This is done by adding padding to the end of sequences that are shorter than the required length. The labels for each intent (tag) are then transformed into numerical representations using LabelEncoder. For example, tags such as "greeting," "jokes," or "curhat" are converted into numerical values such as 0, 1, or 2, respectively, based on their unique order, as illustrated in Table 2.

Table 2. Label Encoding

Tag	Label
'sapaan_awal', 'hobi', 'gaweyan', 'hobi', 'cuaca', 'jenaka	0, 1, 2, 1, 3, 4
'makanan', 'nongkrong', ' curhat', 'nongkrong 2'	5, 6, 7, 8
'fallback', 'hobi', 'temen_pengkhianat', 'arep_dolan'	9, 1, 10, 11

This step is crucial so that the model's output can be compared with the correct labels in a numerical format. This process results in two main components: the input data, which consists of tokenized number sequences, and the output data, which consists of numerical labels, both of which are ready to be used in the training process.

C. Modelling

The model for this chatbot is built using Long Short-Term Memory (LSTM), a type of neural network that has proven to be highly effective in handling sequential data, such as text. LSTM has the ability to retain important information over long periods, enabling the model to understand the context of a conversation more deeply. LSTM is also designed to address the vanishing gradient problem often encountered in traditional neural networks, which can hinder the learning process in models with long data sequences. This capability makes LSTM an ideal choice for applications requiring sustained context understanding, such as chatbots. The LSTM model training process is carried out using a neural network architecture tailored to the characteristics of sequential data. The model is built with a structure consisting of an embedding layer to convert tokens into 128-dimensional vectors, followed by a SpatialDropout1D layer to prevent overfitting, and an LSTM layer with 128 units capable of capturing the context and word sequence in sentences. The output of the LSTM is then processed by a dense layer as the output layer, using the softmax activation function to predict the class of the input. The model is compiled using the sparse_categorical_crossentropy loss function and the Adam optimizer, with accuracy as the evaluation metric.

Fuzzy matching is used to compare the user's input with the list of patterns available in the dataset. The system will accept input in the form of a sentence from the user, which is first processed with fuzzy matching using the fuzzywuzzy library to match the input with patterns in the dataset, ensuring the chatbot remains responsive even in the presence of typos or language variations. Each pattern from every intent is paired with its corresponding tag and stored in a list called pattern_tag_pairs. The get_best_match() function uses the fuzzywuzzy library to calculate the similarity between the user's input and each pattern using the token_set_ratio method. The highest score and its corresponding tag are then returned as the matching result.

The main function, predict_response(), first attempts to match the input using fuzzy matching. The similarity score is then normalized into a percentage value between 0 and 100. If the similarity score exceeds a certain threshold (e.g., ≥ 70%), it is considered sufficiently similar and can be treated as a match, as shown in the example of input strings and pattern strings seen in Table 3.

Table 3. Result of Similiarity Calculation Prediction

Input	Closest Pattern	Levenshtein Distance	Max Lenght	Prediction Tag	Similarity Score
"yo assala mualaikum"	*"assala mualaikum"*	3	18	*sapaan_awal*	84%
"halo brooo"	*"halo bro"*	2	10	*sapaan_awal*	80%
"cuac ane piye bro?"	*"cuaca ne piye?"*	4	19	*cuaca*	79%

Levenshtein Distance will identify the differences between the input string and the pattern, and these differences will be used to calculate the similarity score. After all the scores are calculated, fuzzy matching selects the pattern with the highest score. If the score exceeds a certain threshold (e.g., ≥ 70%), the system directly selects the tag associated with that pattern as the predicted result. Thus, fuzzy matching acts as an initial filter, speeding up and simplifying the process of matching user input to intents, especially when the input is very similar to the training data but may differ in terms of word structure or spelling.

If the similarity score is < 70, it means that the system is not confident enough that the user input closely matches any pattern in the dataset. In this case, fuzzy matching is not used as the main basis for determining the intent. Instead, the task

DOI: 10.34148/teknika.v14i2.1258

of prediction will be handed over to the machine learning model, which is the pre-trained LSTM (Long Short-Term Memory) model. The LSTM model will calculate the highest probability for each intent tag based on the input pattern it has learned during training, as seen in Table 4.

Table 4. Prediction Result If Similiarity < 70

Input	Closest Pattern	Levenshtein Distance	Max Lenght	Prediction Tag	Similarity Score
"aku pengin ketawa"	*"aku pengen ngguyu"*	7	17	*jenaka*	59%
"panganan sing enak ngendi?"	*"panganan enak nang kene apa?"*	13	28	*makanan*	54%
"critakna aku guyonan sing lucu"	*"critakna guyonan"*	13	30	*jenaka*	57%

The result of this input indicates that no sufficiently high similarity value (≥ 70) was found, so the system does not directly take the result from fuzzy matching. In other words, fuzzy matching acts as an initial filter, and if it fails (because the score is < 70), the prediction is then continued by the machine learning model, which is more flexible in understanding complex or unusual sentence patterns. This approach makes the chatbot more adaptive and accurate in handling various types of user input variations.

The chatbot will use predictions based on the LSTM model if the similarity score does not reach the specified threshold. The user's input is converted into a sequence of tokens using the tokenizer, then padded to a uniform length using pad_sequences. This sequence is then fed into the model to produce probabilities for each intent class. If the model's similarity score for the highest prediction is < 40, the chatbot will automatically choose the fallback intent as the default response. If the similarity score is sufficiently high, the label is returned to its text form using the label encoder.

Based on the predicted intent (whether from fuzzy matching or the model), the chatbot retrieves the corresponding list of responses from the dataset and selects one randomly using random.choice(). If the tag is end_conversation, the chatbot also returns a signal to end the conversation. Otherwise, the conversation continues. This approach is a smart combination of fuzzy logic to handle flexible everyday language variations and the LSTM model for pattern-based predictions that have been learned, as well as a safe fallback mechanism when the predictions are uncertain.

D. Evaluation

The model is then compiled using the loss function sparse_categorical_crossentropy, which is suitable for integer labels, along with the Adam optimizer and accuracy metric. The model training is carried out for 200 epochs, and the training results are stored in the history object for further analysis. Based on the training results, the model achieved an accuracy of 99.04% on the training data, indicating that the model has successfully identified important patterns in the dataset with a very high success rate. The recorded loss value of 2.01%, as shown in Figure 4, indicates that the model's error rate is relatively low.

```
# Akurasi dan Loss terakhir (epoch terakhir)
final_accuracy = history.history['accuracy'][-1]
final_loss = history.history['loss'][-1]

print(f"Akurasi : {final_accuracy:}")
print(f"Loss : {final_loss:}")
```

```
Akurasi : 0.9904341697692871
Loss : 0.020135188475251198
```

Figure 4. Training Accuracy and Loss Results

Figure 4 shows that the model has undergone a good training process, with indications that the model did not experience overfitting, which often hinders the model's ability to generalize to unseen data. Performance visualization of the model during training is also performed by plotting the accuracy and loss values as seen in Figure 5. This graph helps to observe whether the model has learned well, whether there are signs of overfitting, and how the model's performance evolves from epoch to epoch.

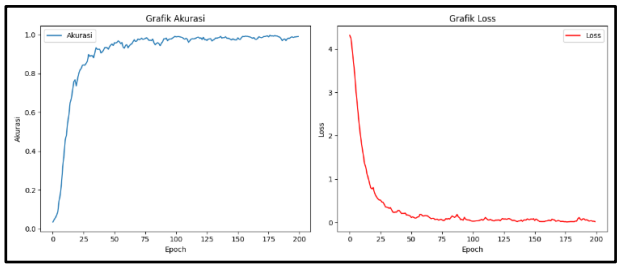


Figure 5. Accuracy and Loss Training Graph

The trained model is saved into the file chatbot_lstm.h5, and other important components such as Tokenizer and LabelEncoder are also saved using pickle. This storage ensures that the model can be reused without the need for retraining and speeds up the prediction process when the chatbot is run. Overall, this pipeline encompasses all essential steps from loading data, processing input, building the model, training, saving, and evaluating the training results so that the LSTM chatbot model is ready for real-world applications. With high accuracy and low loss values, the developed LSTM model demonstrates excellent performance in understanding and responding to conversations in Bahasa Jawa Ngapak. These evaluation results serve as a crucial indicator, proving that the

model has been well-trained and is ready to be used on new data or in real-world conversation situations. This model shows readiness for wide implementation due to its ability to respond accurately and relevantly even in more complex conversational contexts.

E. Deployment

The trained model is stored in .h5 format and integrated into the application using the Flask framework. The chatbot model is deployed by integrating it into the Flask framework as the backend for the application. On the front end, HTML, CSS, and JavaScript are used to design an interactive and responsive user interface. This interface aims to provide an optimal user experience, allowing users to interact directly with the chatbot through a simple and easy-to-use interface, as shown in Figure 6.

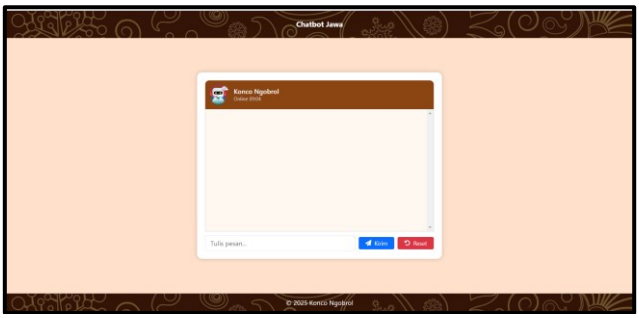


Figure 6. Chatbot Application User Interface

The application user interface is built using HTML and CSS to display the chatbot in an interactive chat box format. The visual design is enriched with a batik background and moving icons, providing a local nuance characteristic of Javanese culture while maintaining a modern look. To enhance the user experience, interactive elements such as an animation loader are added during the prediction process.

Flask is used to process the input provided by the user and return a response in JSON format. In this way, the application can interact directly and in real time with the user. The available functions provide a specific endpoint at the URL /get, allowing users to send a text input message through the URL parameter called msg. This endpoint receives a POST request from the frontend containing the user's input in JSON format (with the key 'msg'). The input is then passed to the predict_response() function, which processes the input and returns the chatbot's response along with the status of whether the conversation should be ended (if the intent is end_conversation). The results are then packaged in JSON format and returned to the frontend to be displayed to the user. The entire Flask application will run in development mode and automatically reload the server if there are changes to the code.

The quality of the chatbot is assessed through testing and evaluation of the chatbot's performance that has been built. The testing is conducted thoroughly using various input scenarios that reflect real conversation contexts. The evaluation covers both quantitative aspects, such as the model's accuracy during training, as well as qualitative aspects based on the relevance and naturalness of the responses provided by the chatbot. The fallback feature is tested to ensure that the system continues to provide polite and appropriate responses, even when receiving input that is not present in the dataset. The results of this testing serve as the basis for evaluating the system's success in understanding and responding to Bahasa Jawa Ngapak effectively in informal conversation contexts.

The testing phase is carried out to evaluate the chatbot's performance in terms of prediction accuracy, response relevance, and its ability to handle variations in user input. The testing is conducted using training data. Based on the training results, the model shows a steadily increasing accuracy from the beginning to the end of the epoch, as seen in the training graph, which indicates improved accuracy and reduced loss. During manual testing with new input, the chatbot is able to provide fairly relevant responses to various types of questions, especially thanks to the combination of fuzzy matching and LSTM. Fuzzy matching plays an important role in handling word variations and minor typos.

If the model is confident enough (≥ 40), the system selects the tag with the highest probability and provides a response based on that tag. Fuzzy matching serves as an initial filter, and if it fails (because the score is < 70), the prediction is continued by the machine learning model, which is more flexible in understanding complex or unusual sentence patterns. This approach makes the chatbot more adaptive and accurate in handling various types of user input variations. The results of the testing can be seen in Figure 7.

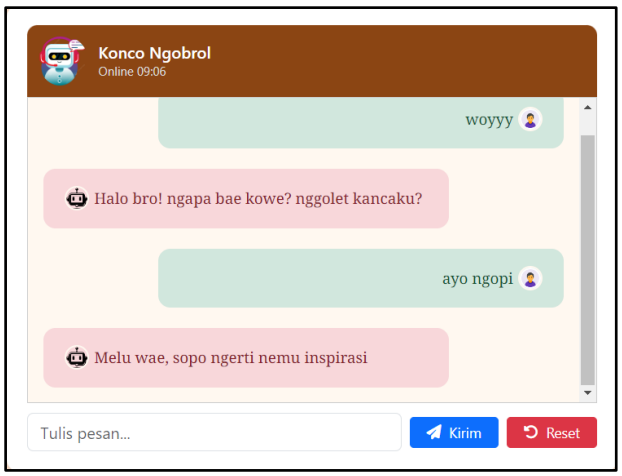


Figure 7. Chatbot Testing Results Display

The results from Figure 7 show that the chatbot system has demonstrated sufficient performance to be used as a casual conversation chatbot or a simple Q&A application. Many consider that this chatbot is not only capable of answering but also serves as a "conversation partner" that reflects the unique personality of the Ngapak community. This indicates that the *Konco Ngobrol* chatbot excels not only technically but also successfully builds an emotional and cultural connection with the users.

The use of the Long Short-Term Memory (LSTM) model in the development of this chatbot offers several advantages

DOI: 10.34148/teknika.v14i2.1258

that align with the sequential and contextual nature of conversational data. The primary strength of LSTM lies in its ability to retain long-term information through an internal memory mechanism known as the *cell state*, making it effective in understanding word sequences and conversational meanings that are not explicitly expressed in a single sentence. This capability is particularly crucial for handling informal conversations, which often involve ellipses, abbreviations, or implied contexts. Furthermore, LSTM demonstrates strong performance in intent classification based on recurring linguistic patterns within the dataset, as evidenced by the model's high accuracy of 99.04% and low loss value.

Nevertheless, LSTM also presents certain limitations that must be taken into account. This model requires relatively longer training times and greater computational resources compared to traditional models, particularly when the dataset size increases significantly. Additionally, the performance of LSTM heavily depends on the quality and diversity of the dataset. In the context of non-standard regional languages such as Ngapak, challenges arise when the model is required to recognize highly variable and unstandardized expressions. To address this issue, the present study incorporates a complementary approach using fuzzy matching, which enhances the system's ability to interpret inputs that are not explicitly represented in the training data.

With a system like this, user interactions become more natural and aligned with the dynamics of everyday communication. This indirectly creates a digital space that reconnects society, particularly the younger generation, with the use of local dialects that are increasingly rare in formal communication media. The system not only represents an artificial intelligence-based technology but also carries a social and cultural mission that is relevant to current societal needs. Furthermore, this chatbot can be utilized in informal educational settings, for entertainment purposes, or even in the development of local platforms based on conversational digital interactions. By integrating modern technology with local wisdom, this study demonstrates that AI development can be directed not only toward technical but also toward supporting the sustainability of cultural identity.

IV. CONCLUSION

This research successfully developed a Javanese Ngapak language chatbot application named *Konco Ngobrol*, using the Long Short-Term Memory (LSTM) model approach to understand and predict user conversation intents. By utilizing a synthetic dataset that reflects the various conversations of young people in the style of Javanese Ngapak and Bahasa Indonesia, the system is designed to handle input contextually, flexibly, and responsively. The training results show that the built LSTM model achieved a high accuracy of 99.04% with a loss rate of 2.01%, indicating that the model was able to learn well from the data without overfitting. The use of fuzzy matching also contributed to increasing the system's robustness against variations in user input, including spelling errors or phonetic variations common in regional languages. The combination of machine learning-based predictions and a rule-based approach in the fallback system makes this chatbot more adaptable to various types of input. For future development, this system can be enhanced by expanding the dataset with real user interaction data, supporting speech input (speech-to-text), and integrating more advanced NLP models like BERT or GPT to improve understanding of more complex contexts for future development in creating a more adaptive and contextual chatbot.

REFERENCES

[1] Arsanti, "Pudarnya Pesona Bahasa Indonesia di Media Sosial (Sebuah Kajian Sosiolinguistik Penggunaan Bahasa Indonesia)," *Ling. Fr. Bahasa, Sastra, dan Pengajarannya*, vol. 4, no. 1, pp. 1–12, 2020, [Online]. Available: http://103.114.35.30/index.php/lingua/article/view/4314

[2] T. R. Gatikasari Mujiastuti, "Identitas Dialek Banyumasan Sebagai Konstruksi Budaya Studi Penggunaan Dialek Banyumasan Di Kalangan Penutur Asli Banyumas Yang Berada Di Semarang," pp. 1–23, 2023.

[3] C. Nugroho and I. P. Kusuma, "Identitas Budaya Banyumasan dalam Dialek Ngapak," *J. Ilmu Komun.*, vol. 21, no. 2, p. 333, 2023, doi: 10.31315/jik.v21i2.4556.

[4] Isrofiah Laela Khasanah and Heri Kurnia, "Melestarikan Budaya Banyumasan Melalui Dialek Bahasa Ngapak," *Kulturist. J. Bhs. dan Budaya*, vol. 7, no. 2, pp. 43–53, 2023, doi: 10.22225/kulturistik.7.2.7135.

[5] A. G. Pawestri, "Membangun Identitas Budaya Banyumasan Melalui Dialek Ngapak Di Media Sosial," *J. Pendidik. Bhs. dan Sastra*, vol. 19, no. 2, pp. 255–266, 2020, doi: 10.17509/bs_jpbsp.v19i2.24791.

[6] E. Adamopoulou and L. Moussiades, *An Overview of Chatbot Technology*, vol. 584 IFIP. Springer International Publishing, 2020. doi: 10.1007/978-3-030-49186-4_31.

[7] H. Jelodar, Y. Wang, R. Orji, and S. Huang, "Deep Sentiment Classification and Topic Discovery on Novel Coronavirus or COVID-19 Online Discussions: NLP Using LSTM Recurrent Neural Network Approach," *IEEE J. Biomed. Heal. Informatics*, vol. 24, no. 10, pp. 2733–2742, 2020, doi: 10.1109/JBHI.2020.3001216.

[8] A. Elcholiqi and A. Musdholifah, "Chatbot in Bahasa Indonesia using NLP to Provide Banking Information," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 14, no. 1, p. 91, 2020, doi: 10.22146/ijccs.41289.

[9] H. Mustafidah, S. Suwarsito, and T. Pinandita, "Natural Language Processing for Mapping Exam Questions to the Cognitive Process Dimension," *Int. J. Emerg. Technol. Learn.*, vol. 17, no. 13, pp. 4–16, 2022, doi: 10.3991/ijet.v17i13.29095.

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780,

1997, doi: 10.1162/neco.1997.9.8.1735.
[11] F. Zakariya, J. Zeniarja, and S. Winarno, "Pengembangan Chatbot Kesehatan Mental Menggunakan Algoritma Long Short-Term Memory," *J. Media Inform. Budidarma*, vol. 8, no. 1, p. 251, 2024, doi: 10.30865/mib.v8i1.7177.
[12] H. A. F. Muhyidin and L. Venica, "Pengembangan Chatbot untuk Meningkatkan Pengetahuan dan Kesadaran Keamanan Siber Menggunakan Long Short-Term Memory," *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 5, no. 2, p. 152, 2023, doi: 10.36499/jinrpl.v5i2.8818.
[13] H. Rumapea, "Deteksi Kemiripan Artikel Melalui Keywords Dengan Metode Fuzzy String Matching Dalam Natural Language Processing," *METHOMIKA J. Manaj. Inform. dan Komputerisasi Akunt.*, vol. 5, no. 1, pp. 60–66, 2021, doi: 10.46880/jmika.vol5no1.pp60-66.
[14] F. Aulia, A. Farisi, R. S. Perdana, P. P. Adikara, U. Brawijaya, and P. Korespondensi, "Klasifikasi Intensi Dengan Metode Long Short-Term Memory Intent Classification With Long Short-Term Memory Method in Indonesian Language Chatbot," vol. 11, no. 5, pp. 1051–1058, 2024, doi: 10.25126/jtiik.2024118000.
[15] A. Fariz Zulhilmi, R. Setya Perdana, and P. Korespondensi, "Pengenalan Entitas Bernama Menggunakan Bi-Lstm Pada Chatbot Bahasa Indonesia Named Entity Recognition Using Bi-LSTM in Indonesian Language Chatbot," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 10, no. 7, pp. 1425–1430, 2023, doi: 10.25126/jtiik.2024117968.
[16] P. B. Wintoro, H. Hermawan, M. A. Muda, and Y. Mulyani, "Implementasi Long Short-Term Memory pada Chatbot Informasi Akademik Teknik Informatika Unila," *Expert J. Manaj. Sist. Inf. dan Teknol.*, vol. 12, no. 1, p. 68, 2022, doi: 10.36448/expert.v12i1.2593.
[17] F. Deby Fambayun, G. Asrofi Buntoro, and F. Masykur, "Penerapan Algoritma Neural Network Pada Chatbot Bahasa Jawa Tingkat Tutur Krama Alus," *J. Tek. Inform.*, vol. 14, no. 1, pp. 40–46, 2022.
[18] Zhaozhen Xu, "Prevent the Vanishing Gradient Problem with LSTM," Baeldung. Accessed: Jan. 07, 2025. [Online]. Available: https://www.baeldung.com/cs/lstm-vanishing-gradient-prevention
[19] Ethan Nam, "Understanding the Levenshtein Distance Equation for Beginners," medium.com. Accessed: May 01, 2025. [Online]. Available: https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0
[20] M. S. M. Rudwan and J. V. Fonou-Dombeu, "Hybridizing Fuzzy String Matching and Machine Learning for Improved Ontology Alignment," *Futur. Internet*, vol. 15, no. 7, 2023, doi: 10.3390/fi15070229.
[21] H. Gadge, "A Chatbot for Medical Purpose using Deep Learning," vol. 10, no. 05, pp. 441–447, 2021.