ILKOM Jurnal Ilmiah Vol. 16, No. 1, April 2024, pp. 59-67
Accredited 2nd by RISTEKBRIN No. 200/M/KPT/2020; E-ISSN 2548-7779 | P-ISSN 2087-1716

DIET Classifier Model Analysis for Words Prediction in Academic Chatbot

Wistiani Astuti [a,1,*]; Aji Prasetya Wibawa [b,1]; Haviluddin [c,1] ; Herdianti Darwis [a,2]
[a] *Universitas Muslim Indonesia, Jln. Urip Sumoharjo Km. 5, Makassar, 90231, Indonesia*
[b] *Universitas Negeri Malang, Jln. Urip Sumoharjo Km. 5, Makassar, 90231, Indonesia*
[c] *Universitas Mulawarman, Jln. Kuaro No.1, Samarinda, 75123, Indonesia*
[1] *wistiani.astuti@umi.ac.id;* [2] *aji.prasetya.ft@um.ac.id;* [3] *haviluddin@gmail.com;* [4] *herdianti.darwis@umi.ac.id*
** Corresponding author*

Article history: Received February 15, 2023; Revised April 02, 2023; Accepted December 08, 2023; Available online April 26, 2024

Abstract
One prevalent conversational system within the realm of natural language processing (NLP) is chatbots, designed to facilitate interactions between humans and machines. This study focuses on predicting frequently asked questions by students using the Duel Intent and Entity Transformer (DIET) Classifier method and assessing the performance of this method. The research involves employing 300 epochs with an 80% training data and 20% testing data split. In this study, the DIET Classifier adopts a multi-task transformer architecture to simultaneously handle classification and entity recognition tasks. Notably, it possesses the capability to integrate diverse word embeddings, such as BERT and GloVe, or pre-trained words from language models, and blend them with sparse words and n-gram character-level features in a plug-and-play manner. Throughout the training process of the DIET Classifier model, data loss and accuracy from both training and testing datasets are monitored at each epoch. The evaluation of the text classification model utilizes a confusion matrix. The accuracy results for testing the DIET Classifier method are presented through four case studies, each comprising 25 text messages and 15 corresponding chatbot responses. The obtained accuracy values range from 0.488 to 0.551, F1-Score values range from 0.427 to 0.463, and precision range from 0.417 to 0.457.

Keywords: Chatbot; Confusion Matrix; Diet Classifier.

Introduction

The evolution of Natural Language Processing (NLP) has significantly contributed to the proliferation of chatbot technology, which serves as a pivotal interface facilitating interactions between humans and machines. Chatbots are designed to engage users in dialogue, leveraging advanced algorithms to swiftly respond to diverse inquiries. Integral to their efficacy is the capacity to retain and recall information seamlessly, thereby enhancing user satisfaction. Moreover, chatbots demonstrate proficiency in information retrieval, adeptly sourcing and disseminating relevant data. This amalgamation of capabilities underscores the utility and effectiveness of chatbots in augmenting human-computer interaction paradigms [1]. Among several types of chatbots, the primary category is commonly referred to as virtual assistants, designed to cater to users' needs across various domains and sectors [2]. NLU presents a formidable subdomain within the field of NLP, tasked with the extraction and comprehension of specialized conversational nuances. Employing sophisticated algorithms, NLU endeavors to condense natural language into a structured ontology, facilitating the extraction of pertinent entities inherent to the linguistic context [3]. In the realm of statistics, prediction constitutes a fundamental component of inferential statistics, focusing on forecasting future values based on past events. Data analysis frequently encounters incomplete datasets due to the presence of missing data, attributed to various factors such as equipment malfunction, human error, natural disasters, or unidentified causes. The Rasa framework emerges as an artificial intelligence (AI) infrastructure characterized by expansiveness and flexibility within the AI framework.

Rasa NLU stands as an open-source NLP library tailored for the classification of intents and extraction of entities within chatbots, thereby aiding in the construction of NLP systems. Within the Rasa ecosystem, two primary components are discernible: Rasa NLU and Rasa Core [4]. Several research studies pertaining to chatbots, such as [5] Several research endeavors have focused on the comprehensive analysis of the open-source framework Rasa, culminating in the conclusion that chatbots built upon Rasa possess superior capabilities compared to other open-source alternatives. Notably, the Dual Intent and Entity Transformer (DIET) architecture, integrated within Rasa Open Source 1.8.0, leverages trained embeddings within the Rasa NLU pipeline. DIET represents a multitask transformer architecture capable of simultaneous intent classification and entity recognition. Comprising multiple components, DIET offers flexibility for the interchange of diverse elements. A prominent feature includes its ability to merge different word embeddings such as BERT and GloVe, or trained word embeddings from language models, with infrequently occurring words and character-level n-gram features in a plug-and-play fashion. Many pretrained language

models entail significant computational resources and inference time, rendering them unsuitable for conversational AI applications. In contrast, DIET embodies a modular architecture that empowers software developers with enhanced flexibility in experiments, matching pretrained language models in accuracy while outperforming the current state-of-the-art (SOTA) and training them six times faster [6].

Research [4] have employed chatbots for job vacancy information based on WhatsApp using NLP techniques. The implementation of such applications facilitates job seekers in efficiently finding employment opportunities that align with their desired criteria. Research [5] conducting literature reviews and analyses on the open-source chatbot framework, Rasa, have concluded that Rasa-based chatbots exhibit superior capabilities compared to other open-source alternatives. The researchers [7] utilized a WhatsApp bot as a COVID-19 statistics provider, employing PHP, Flask, and MySQL in their implementation. They developed a chatbot application accessible via WhatsApp, offering information and statistical data on COVID-19 in Indonesia. The researchers [8] employed a chatbot to bolster healthcare services for users, enabling swift responses to patients experiencing accidents or chronic illnesses. Their framework design encompasses four levels: data, information, and service levels, aimed at enhancing the efficiency and effectiveness of healthcare delivery. Researchers [9] have provided healthcare services for individuals suffering from obesity and prediabetes through the "Tess" chatbot. Supported by artificial intelligence, Tess can attend to patients in large groups and interact outside of medical professionals' working hours. Researchers [10] have employed a chatbot to mitigate mental health issues among adolescents. By presenting users with a series of questions, the chatbot identifies users' emotions and calculates the percentage of negativity within the conversation. Furthermore, the chatbot classifies the user's mental status into three types: normal, stressed, or depressed. Researchers [11] utilize chatbots to assist industry players or small businesses by automating customer support functions, thereby enhancing profitability. Researchers [12] have developed a chatbot aimed at enhancing a particular hotel's services by focusing on recommending available amenities and facilitating user access to the hotel anytime and anywhere. Meanwhile, the researchers [13] have created a chatbot implemented within a smartphone application for automated online shopping. Offering round-the-clock service, it responds to customers with general inquiries, calculations, and checks on product availability in the online store. Researchers [14] utilize a chatbot for providing information about Covid-19 utilizing the DIET Classifier model. The presence of this chatbot facilitates convenience for the public regarding the Covid-19 outbreak.

This study employs a chatbot utilizing the Rasa framework to predict the intent and entities related to frequently asked questions by students in academic fields using the DIET Classifier model. There are 10 intents utilized in the chatbot's prediction, including greetings, negations, expressions of gratitude, scheduling inquiries, grade inquiries, payment queries, examination-related queries, graduation inquiries, registration inquiries, community service program inquiries, and internship inquiries. By analyzing the prediction results of questions and answers within the chatbot, essential information required by students can be provided, thereby facilitating the prediction of frequently asked questions related to academic information.

Method

A. Chatbot Architecture

Chatbots are generally constructed with two primary components: information extraction and dialog management. The information extraction component is utilized to identify the intent, which is the process of determining the purpose of the message inputted by the user. Dialog management, on the other hand, is employed to handle the conversational needs between the chatbot and the user. In **Figure 1** depicting the architecture of a chatbot, the Finite State Machine (FSM) constitutes a modeling aspect present in computer systems such as chatbots, aimed at determining the actions to be taken by the chatbot machine when the user inputs a question. By utilizing FSM, the chatbot machine can be programmed to execute actions or provide responses to the user's queries. FSM comprises two main keywords: state and action. State represents a condition inputted by the user, which is directed to the chatbot machine, while action denotes a task to be performed by the chatbot machine while in a specific state.

The development of a chatbot entails representing the domain of the chatbot as fundamental knowledge and as part of the learning environment. The domain encompasses various types of intents, actions, and templates for responses to user messages. In this study, the chatbot is developed to provide information related to community service program (KKN) procedures, fees, registration, schedules, grades, internships, examinations, and graduation to students. The data utilized for constructing the chatbot include intents, entities, and sample dialogs. The Rasa framework is represented in two parts: Rasa NLU and Rasa Core. Rasa NLU handles the Natural Language Understanding (NLU) process, while Rasa Core manages the dialog management within the Rasa framework. **Figure 2** illustrates the modeling process of the chatbot, where the initial domain is based on question and answer data. The domain specifies the training data that generates a model for the chatbot. The quality of training data can be enhanced to produce a chatbot model capable of effectively responding to user information needs. Each question inputted by the user to the bot is processed through the DIET Classifier model, resulting in bot responses as specified in the stories.md file.

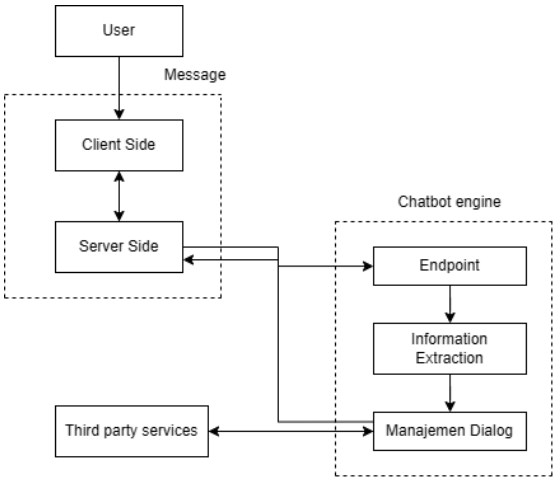

Figure 1. Chatbot Arcitecthure

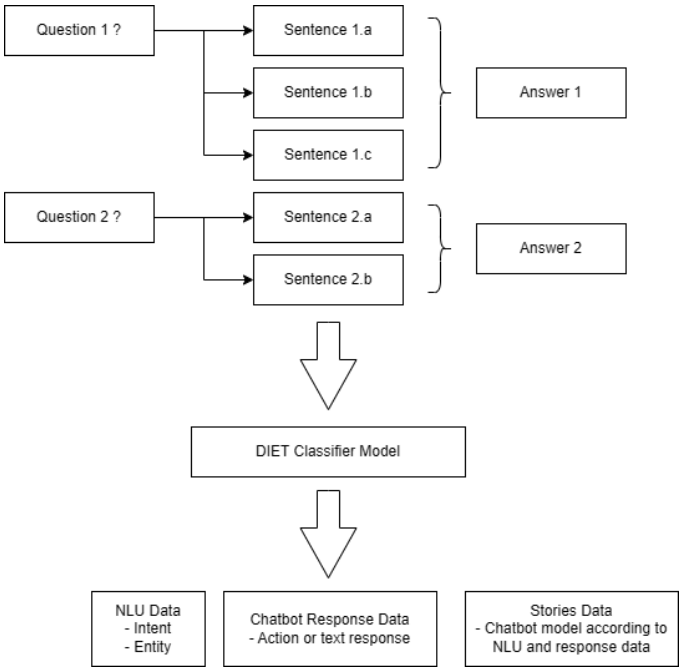

Figure 2. Chatbot Process Modeling

B. RASA Natural Language Understanding

Rasa NLU serves as a tool for constructing conversational systems, particularly in the domain of Natural Language Understanding (NLU), and represents an open-source module within natural language processing. The modules of Rasa NLU are integrated with various natural language processing and machine learning processing libraries within a consistent API [15]. Rasa predicts a set of slot labels and slot values associated with different segments of input rather than a sequence of slots for each word input [16]. Within Rasa NLU, a pipeline is required to process text along with other components. Pipelines such as spacy_sklearn begin by processing text that is tokenized and annotated with part-of-speech (POS) tags using the spaCy NLP library. Subsequently, the spaCy features search for GloVe vectors for each token and combine them to create a representation of the entire sentence. Following this, scikit-learn classification is performed to train words from the dataset where, by default, vectors are supported with multi-class classifiers trained with validation [15]. **Table 1** are some of the commands contained in RASA NLU [17].

Table 1. Commands In RASA NLU[17]

Code	Function
Rasa init	Create new projects such as training data, configuration files, and actions
Rasa train	Train the model using data from NLU and save the trained model in ./models
Rasa interactive	Create new training data to start an interactive learning session
Rasa shell	Loading your trained model

Code	Function
Rasa run	Start server with your trained model
Rasa run actions	Starting an action server using the Rasa SDK
Rasa visualize	Produce a visual representation of your story.
Rasa test	Test the trained Sense model on any file starting with test_
Rasa data split nlu	Performing an 80/20 split of your NLU training data
Rasa data convert	Convert training data between different formats
Rasa data validate	Checking for inconsistencies in domain, NLU, and conversation data.
Rasa export	Export conversations from tracker store to event broker
Rasa x	Launching Rasa X in local fashion
Rasa -h	Display all available commands

1) Pipeline Chatbot RASA NLU

RASA NLU features an entity retrieval module, facilitating the development of university academic information request chatbots. The interaction between the bot and the user occurs via the RASA or RASA X framework, where users seek information and messages are sent to the bot. The RASA NLU abstract entities from these messages, allowing the bot to derive the user's intent and respond appropriately using the RASA NLU interpreter engine. Data is organized into several sections such as synonyms and regex features. **Figure 3** illustrates an example of the pipeline principle in RASA NLU.

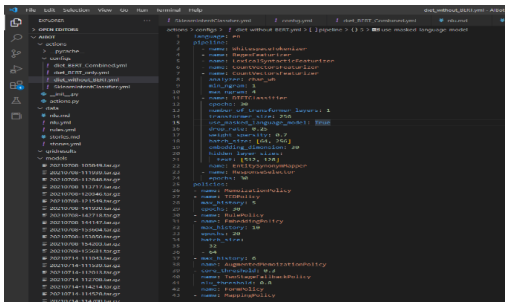

Figure 3. The pipeline principle RASA NLU

C. DIET (Dual Intent and Entity Transformer) Classifier

DIET represents a novel multitask architecture for classification and entity recognition, distinguished by its ability to integrate pre-trained word embeddings from language models with infrequently occurring words and character-level n-gram features in a plug-and-play mode. DIET, even without pre-trained embeddings, can enhance complex NLU datasets. Moreover, incorporating pre-trained word and sentence embeddings from language models further improves overall accuracy across all tasks.[18]. The representation of the Diet architecture in **Figure 4** consists of several key components. The sentence "main ping pong" conveys the intent of playing a game and contains an entity named "ping pong" with its associated value. The weights from the feed-forward layer are shared among tokens.

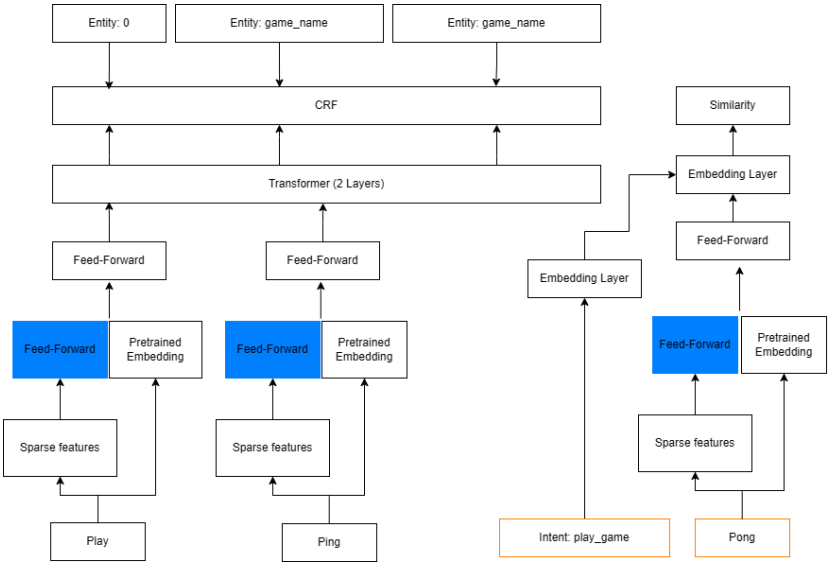

Figure 4. DIET architecture [18]

The most important parts of the Diet Classifier architecture are:

a) The uniqueness lies in treating input sentences as sequences of tokens, which can be words or subwords, depending on the featureization pipeline. [19] In addition, a special classification token is appended at the end of each sentence. Rarely used features are encoded using one-hot token level and multi-hot character n-grams ($n \leq 5$). Character n-grams contain abundant information, hence to avoid overfitting, dropout is applied to these rare features. Dense features typically become embeddings with any pre-trained word like ConveRT. [20], BERT[19], or GloVe [21]. ConveRT is also trained as a sentence encoder, and when ConveRT is set as the initial embedding for the CLS token, the sentence encoding will be obtained from ConveRT. This will add contextual information to the entire sentence in addition to information from word embeddings.

b) Transformers are needed to encode context across complete sentences, and use 2-layer transformers[22] with attention to relative position [23]. Transformers require input to be of the same dimension as the transformer layers, thus the combined features will be passed through other fully connected layers with the same weights.

c) The sequence of entities is predicted through Conditional Random Field (CRF) tagging on the layer above the transformer sequence corresponding to the input token sequence. CRFs and sequence labeling also play a significant role in Named Entity Recognition (NER) tasks. With this model, text is trained to understand its structure and meaning. [2].

d) Intent Classification involves output transformation for the CLS token and label from the transformer, which are stored in a single semantic vector space. During inference, the similarity of points serves as an evaluator for all possible labels intended. [24].

e) Masking serves the purpose of additional training by predicting randomly masked input tokens. [20], [24], [6], [25].

Results and Discussion

In the creation of a chatbot, an NLU pipeline is used to script or encode several files that will be utilized by the chatbot, such as the creation of intents in nlu.md, stories.md, entities in domain.yml, and config.yml. The training data for NLU in this research consists of sample sentences, where each sentence has been labeled with intent or entity names in the nlu.md pipeline. Each intent contains multiple samples in the form of different sentences with the same meaning. In addition to creating intents, a dialogue model is also constructed and organized based on intent and entity labels in the NLU data. The dialogue model, or alternatively referred to as stories, will contain sample dialogues consisting of intent types and responses from the chatbot, as depicted in **Figure 5**. In **Figure 5**, phrases like ##greeting up to utter_greeting will provide dialogue responses from the chatbot for greeting types or salutations entered by the user to the bot, such as hello, assalamualaikum, hi, and so on. Similarly, phrases like ##confirmation up to utter_confirmation will also provide confirmation responses such as yes, ok, please, and so on.

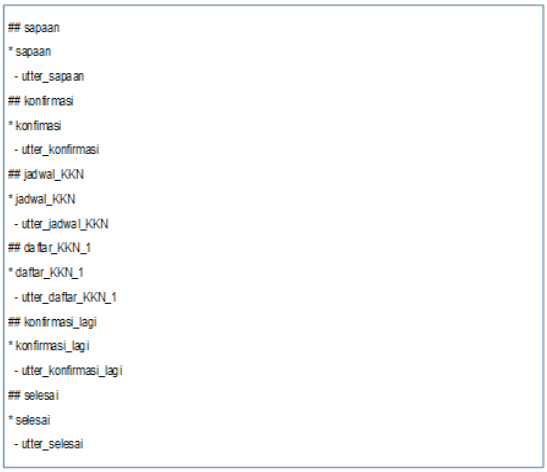

Figure 5. Pipeline NLU stories.md

NLU requires mapping to create a flow for the chatbot. Designing a simple chatbot flow will adapt to the questions and answers found in intents and entities, and after designing a simple chatbot, it will serve several conversational purposes that require more complex NLU designs. Several question and answer designs for academic requirements such as information on community service program (KKN) registration requirements, types of KKN, exams, and so on.

In the chatbotflow for KKN registration requirements, there are question designs that will be asked to the bot and subsequently, the bot will respond with text according to the NLU training data. In this part of the chatbot flow for KKN registration requirements, users do not need to log in, but they only need to greet the bot, and then the bot will provide response answers according to the questions inputted by the user regarding KKN registration requirements. For

example, in **Figure 6**, where the user initially greets the bot with the word "Hello," and the bot will respond to the user's greeting with the word "Hello too."

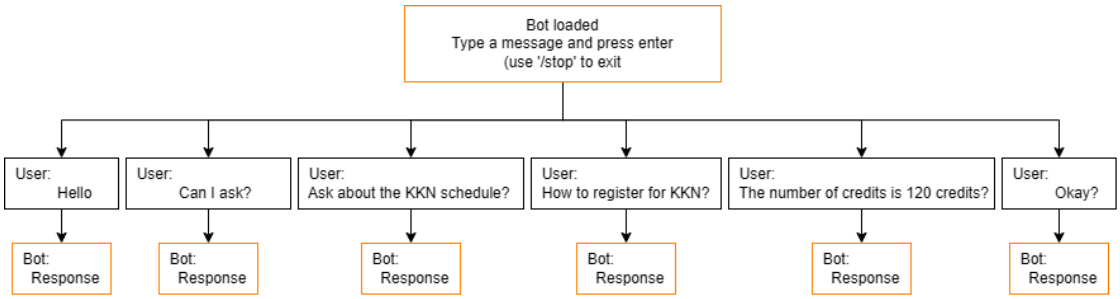

Figure 6. Chatbotflow KKN Registration Requirements

The chatbot flow for KKN types will provide questions as shown in **Figure 7**, aiming to offer comprehensive information regarding the types of KKN available at UMI. For instance, users can inquire from the bot about the number of KKN types present on the UMI campus. The bot will respond to the user's query by stating that there are 3 types of KKN: thematic KKN, professional KKN, and PPMD KKN, along with an explanation of each type of KKN.

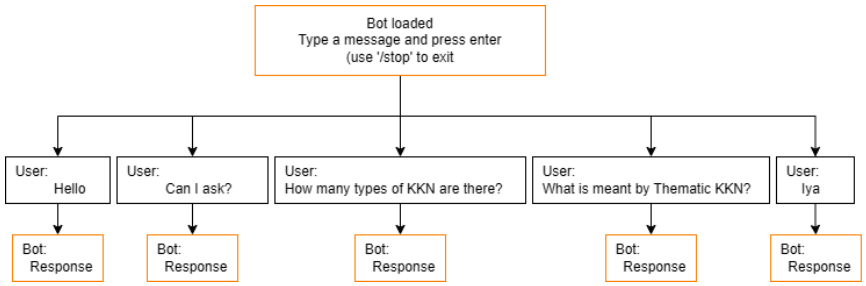

Figure 7. Chatbotflow KKN Type Information

This study utilizes a model with weights based on a multilingual cased BERT base, incorporating various parameters such as the number of transformer layers, transformer size, batch size, weight sparsity, hidden layer size, and others. **Figure 8** illustrates the block diagram for the framework of the DIET classifier. In this block diagram, the task of the interpreter or the text message "When is the implementation of KKN?" is to convert the input text into structured data in the Rasa NLU pipeline, which is then passed on to the dialog management component in Rasa Core. In Rasa Core, the input text is checked using the DIET Classifier model, and a response is provided once the output entity has been verified and matches the input entity. The block diagram for the DIET Classifier model depicted in **Figure 8** is utilized to observe the flow or operation of the model in Rasa Core. **Figure 8** comprises three parts used to generate responses consistent with the chatbot's responses within intents, namely Rasa NLU, Rasa Core, and Rasa NLG. Rasa NLU includes Intent Classification and Entity Extraction. In intent classification, each word or sentence inputted by the user to the chatbot is classified accordingly. All words or sentences created within the intent are classified accordingly, such as the greeting intent. The greeting intent contains words related to greeting the chatbot, such as hello, hey, hi, halo, assalamualaikum, and so on. Entities represent structured information within user messages. Entity extraction functions to determine training data to train the DIET Classifier model.

In the block diagram, Rasa Core functions to process all intents and entities provided by Rasa NLU. The DIET Classifier model is employed in the block diagram when the input entity begins to fetch data from Rasa NLU. The output of the DIET Classifier model in Rasa Core is sent to the action mask to perform actions, followed by a normalization process to check if the data inputted by the user in Rasa NLU matches the trained intents. After normalization, the data is passed on to the sample action as a response from the chatbot to the user's input, and the response data is sent to the entity output to check the deepest entity in the training data. From the entity output, the response data is sent to the action type listed in the domain.yml file. The action type is then cross-checked with the input entity to match the user's input question before the data is sent to the user as a response from the chatbot. After the matching process and alignment with the chatbot's response and training data found in nlu.md and domain.yml, the response will be displayed to the user as the chatbot's response to the user's input question.

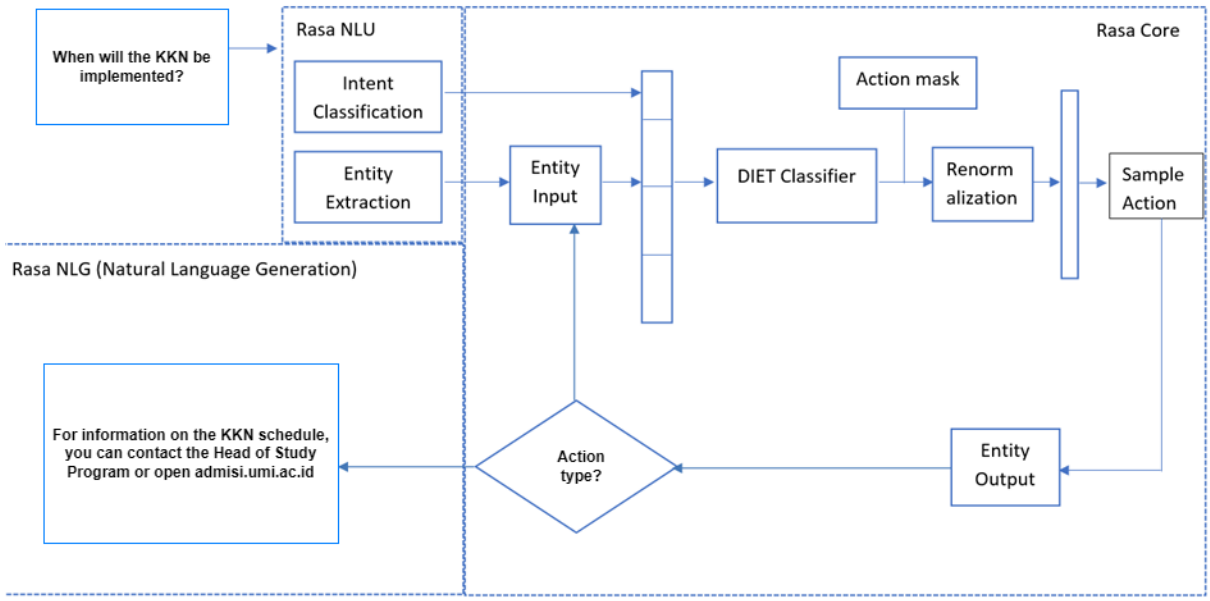

Figure 8. DIET Classifier Model Combination Block Diagram

The testing of the DIET Classifier model involves two conditions. In the first condition, testing is conducted using various parameters such as the number of transformer layers, transformer size, use of masked language model, dropout rate, weight sparsity, embedding dimension, batch size, and hidden layer sizes in the pipeline, which are also utilized in the BERT model testing. In the second condition, testing is performed without utilizing the pipeline components used in the first condition but instead using only the white space tokenizer, lexical syntactic featurizer, count vectors featurizer, analyzer, min_ngram, max_ngram, and entity synonym mapper in the pipeline. Both conditions of DIET Classifier testing employ the same policies. The difference between the two DIET Classifier tests lies in the parameters used. The function of the number of transformer layers in the first condition is to determine the number of encoder layers, whereas the second condition does not use the number of transformer layers parameter. The pipeline with the name DIET Classifier does not depend on other models like in the BERT pipeline. By default, the RASA NLU configuration for the DIET Classifier pipeline includes the white space tokenizer, regex featurizer, lexical syntactic featurizer, count vector featurizer, analyzer, min_ngram, max_ngram, epochs, entity synonym mapper, and response selector. In RASA Core, the pipeline policies use the memoization policy, TED policy with max history and epochs, and rule policy. The parameter for the TED policy's max history is utilized to control how much dialog history the model will consider to decide the next action to take.

The evaluation of question and answer interactions between the chatbot and users, where the questions and answers correspond to the data in nlu.md and domain.yml, yielded results from the DIET Classifier model with parameters including the number of transformer layers, transformer size, dropout rate, use of masked language model, weight sparsity, batch size, and hidden layers, as displayed in **Figure 9** for the intent confusion matrix. Table 2 presents the results of the intent confusion matrix for the scenario of testing questions and answers that align with the training data in nlu.md and domain.yml. Each user input provided to the chatbot in this DIET Classifier model is initialized with intents and entities present in the training data, resulting in intents representing entities receiving a value of 1 compared to intents without entities, which may vary based on the number of repeated questions and answers provided by the chatbot.

A1. Schedule	1	0	0	0	0
A2. InternshipList	0	1	0	0	0
A3. Confirmation	0	0	2	0	0
A4. Greetings	0	0	0	3	0
A5. Finish	0	1	0	0	2
	A1	A2	A3	A4	A5

Figure 9. Intent Confusion Matrix

The accuracy results of the DIET Classifier model are presented in **Table 2**, exhibiting slight variations across several case studies. The accuracy, F1-score, and precision for both train and test datasets may fluctuate depending on the number of user questions and corresponding chatbot responses aligned with the training data in nlu.md and

domain.yml. Meanwhile, **Figure 10** illustrates the distribution of confidence levels for various intents tested using the DIET Classifier model. Green indicates frequently asked questions by students to the chatbot, with responses aligned with the chatbot's answers to the user. Conversely, red highlights instances of errors or discrepancies between the user's input questions and the chatbot's responses, typically resulting from user input errors where the intent is not present in the training data.

Table 2. Accuracy results of 4 cases with DIET Classifier Model

Experiment	Train Accuracy	Train F1-Score	Train Precision	Test Accuracy	Test F1-Score	Test Precision
Study Case 1	0.981±0.019	0.973±0.027	0.968±0.032	0.528±0.048	0.435±0.068	0.417±0.075
Study Case 2	0.825±0.095	0.802±0.088	0.813±0.060	0.488±0.088	0.427±0.096	0.442±0.127
Study Case 3	0.921±0.041	0.896±0.050	0.885±0.051	0.512±0.128	0.463±0.129	0.457±0.123
Study Case 4	0.922±0.002	0.905±0.001	0.899±0.001	0.551±0.089	0.462±0.101	0.434±0.112

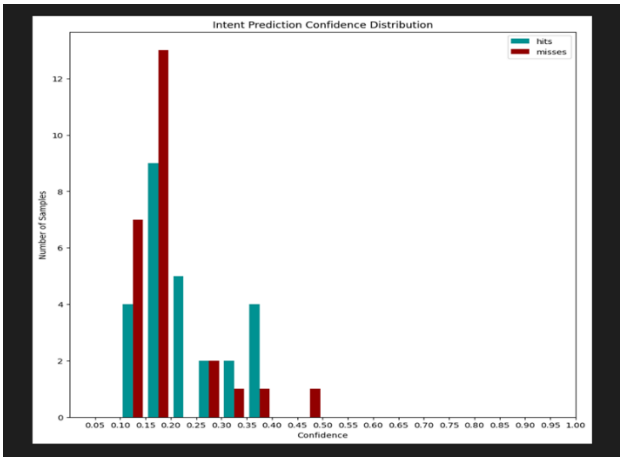

Figure 10. Intent Prediction Confidence Distribution

Conclusion

In this study, the dataset utilized pertained to a closed domain encompassing solely academic information, payment transactions, and Community Service Program (KKN) within the Computer Science faculty, utilizing the RASA Open Source framework with model testing conducted using the DIET Classifier. The training data was divided into two components: Rasa NLU and Rasa Core. The NLU training data consisted of 10 intents, while the dialogue training data was structured based on the anticipated intents likely to be received by the chatbot and the corresponding text responses. The text processing conducted within the chatbot utilizing the DIET Classifier model to interpret user text messages yielded promising results, although the quantity of dialogue training samples and testing data may require augmentation and refinement to ensure optimal model performance. Evaluation outcomes of the DIET Classifier model demonstrated the chatbot's proficient comprehension of user text messages. The study results exhibited varying accuracy levels across the four tested case studies. Across these studies, out of 25 text messages and 15 bot responses, the accuracy rates ranged from 0.488 to 0.551, F1-Score values ranged from 0.427 to 0.463, and precision values ranged from 0.417 to 0.457. Future research endeavors are encouraged to explore the utilization of BERT models with larger training datasets and a more extensive rule base, potentially leading to significantly improved accuracy, F1-Score, and precision outcomes.

References

[1] D. Biswas, “Self-improving Chatbots based on Reinforcement Learning Self-improving Chatbots based on Reinforcement Learning,” no. May, 2019.

[2] A. Jiao, “An Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network,” *J. Phys. Conf. Ser.*, vol. 1487, no. 1, 2020, doi: 10.1088/1742-6596/1487/1/012014.

[3] P. Lauren and P. Watta, “A Conversational User Interface for Stock Analysis,” *Proc. - 2019 IEEE Int. Conf. Big Data, Big Data 2019*, pp. 5298–5305, 2019, doi: 10.1109/BigData47090.2019.9005635.

[4] S. Raj, *Building Chatbots with Python*. 2019.

[5] R. K. Sharma, “An Analytical Study and Review of open S ource Chatbot framework , R ASA,” vol. 9, no.

06, pp. 1011–1014, 2020.

[6] T. Bunk, D. Varshneya, V. Vlasov, and A. Nicho, "DIET: Lightweight language understanding for dialogue systems," *arXiv*, 2020.

[7] D. A. N. Mysql, "Bot Whatsapp Sebagai Pemberi Data Statistik," vol. 1, no. 2, pp. 282–293, 2020.

[8] K. Chung and R. C. Park, "Chatbot-based heathcare service with a knowledge base for cloud computing," *Cluster Comput.*, vol. 22, pp. 1925–1937, 2019, doi: 10.1007/s10586-018-2334-5.

[9] T. N. Stephens, A. Joerin, M. Rauws, and L. N. Werk, "Feasibility of pediatric obesity and prediabetes treatment support through Tess, the AI behavioral coaching chatbot," *Transl. Behav. Med.*, vol. 9, no. 3, pp. 440–447, 2019, doi: 10.1093/tbm/ibz043.

[10] F. Patel, R. Thakore, I. Nandwani, and S. K. Bharti, "Combating depression in students using an intelligent ChatBot: A cognitive behavioral therapy," *2019 IEEE 16th India Counc. Int. Conf. INDICON 2019 - Symp. Proc.*, pp. 1–4, 2019, doi: 10.1109/INDICON47234.2019.9030346.

[11] R. Singh, M. Paste, N. Shinde, H. Patel, and N. Mishra, "Chatbot using TensorFlow for small Businesses," *Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2018*, no. Icicct, pp. 1614–1619, 2018, doi: 10.1109/ICICCT.2018.8472998.

[12] F. P. Putri, H. Meidia, and D. Gunawan, "Designing intelligent personalized chatbot for hotel services," *ACM Int. Conf. Proceeding Ser.*, pp. 468–472, 2019, doi: 10.1145/3377713.3377791.

[13] A. Nursetyo, D. R. I. M. Setiadi, and E. R. Subhiyakto, "Smart chatbot system for E-commerce assitance based on AIML," *2018 Int. Semin. Res. Inf. Technol. Intell. Syst. ISRITI 2018*, pp. 641–645, 2018, doi: 10.1109/ISRITI.2018.8864349.

[14] W. Astuti, D. P. I. Putri, A. P. Wibawa, Y. Salim, Purnawansyah, and A. Ghosh, "Predicting Frequently Asked Questions (FAQs) on the COVID-19 Chatbot using the DIET Classifier," Apr. 2021, doi: 10.1109/EIConCIT50028.2021.9431913.

[15] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open source language understanding and dialogue management," *arXiv*, pp. 1–9, 2017.

[16] T. Desot, S. Raimondo, A. Mishakova, F. Portet, and M. Vacher, "Towards a french smart-home voice command corpus: Design and NLU experiments," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11107 LNAI, pp. 509–517, 2018, doi: 10.1007/978-3-030-00794-2_55.

[17] A. Singh, K. Ramasubramanian, S. Shivam, A. Singh, K. Ramasubramanian, and S. Shivam, *Introduction to Microsoft Bot, RASA, and Google Dialogflow*. 2019.

[18] V. A. Bhagwat, "Deep Learning for ChatBots," *Scholarworks.Sjsu.Edu*, p. 56, 2018, [Online]. Available: https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1645&context=etd_projects.

[19] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, pp. 4171–4186, 2018.

[20] M. Henderson *et al.*, "A repository of conversational datasets," *arXiv*, 2019, doi: 10.18653/v1/w19-4101.

[21] and C. Jeffrey Pennington, Richard Socher and Manning., "No Title," in *Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[22] Ł. K. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, "Attention Is All You Need Ashish," 2017, doi: 10.1109/2943.974352.

[23] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 2, pp. 464–468, 2018, doi: 10.18653/v1/n18-2074.

[24] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "StarSpace: Embed All The Things!," 2018.

[25] V. Vlasov, J. E. M. Mosig, and A. Nichol, "Dialogue Transformers," pp. 1–8.