Smart College Chatbot Using RASA Framework

Sneha Nadig

B Computer Science and Engineering
Bangalore institute of technology snehanadig07@gmail.com

Sunayana Prasanna Rao

Computer Science and Engineering
Bangalore institute of technology sunayanaprao@outlook.com

Spurthi Bhaskar Vernekar

Computer Science and Engineering
Bangalore institute of technology spurthivernekar24@gmail.com

Meghana N

Computer Science and Engineering
Bangalore institute of technology meghanan098@gmail.com

N. Thanuja
Assistant Professor
Computer Science and Engineering
Bangalore institute of technology

Abstract

Chatbots are increasingly being utilized to help human performance to boost communication ease and provide better and faster services. The Smart College Assistant, developed using Rasa's advanced natural language understanding (NLU) and dialogue management capabilities, signifies a significant advancement in conversational AI tailored for the education sector. This assistant provides students and parents with a sophisticated and user-friendly interface to effortlessly access information on admissions, program details, campus events, and departmental guidance. Leveraging Rasa's flexibility for continuous improvement through real user interactions and feedback, the assistant evolves over time to better understand and respond to user needs, ultimately transforming the way individuals engage with academic resources and support services. By offering a personalized and intuitive virtual companion, the Smart College Assistant exemplifies the potential of AI-driven solutions to enhance user experiences and streamline access to educational information and services.

Keywords—chatbot, rasa framework, deep learning, DIET, transformer model

I. INTRODUCTION

The Smart College Assistant, developed using Rasa, represents a cutting-edge application of conversational AI technology aimed at improving the academic experience for students and parents. Powered by Rasa's robust natural language understanding (NLU) capabilities, this virtual assistant excels in interpreting and responding to a wide range of queries related to admissions, academic programs, campus resources, events, and administrative procedures. Rasa's open- source framework provides the necessary tools to build and deploy sophisticated chatbots, enabling accurate and context- aware interactions that mimic natural human conversations.

At its core, the Smart College Assistant leverages machine learning techniques embedded within Rasa to process and comprehend natural language queries with high accuracy. This enables seamless communication in everyday language, allowing users to engage with the assistant as if they were interacting with a human advisor. The assistant's NLU capability is crucial for understanding and extracting key information from user messages, ensuring that queries are addressed promptly and effectively.

Rasa Framework has its special effect where a trained model is used for the prediction and generating responses based on the predicted intent and entities accurately.

This empowers the assistant to handle complex conversations, maintain context across multiple interaction turns, and provide personalized guidance tailored to individual needs. By leveraging dialogue policies and machine learning algorithms, the assistant

can navigate intricate procedures and offer step-by-step assistance.

Rasa's flexibility and extensibility are instrumental in enabling continuous improvement and adaptation of the Smart College Assistant. Through ongoing training using real user interactions and feedback, the assistant evolves to better understand user intents and refine its responses over time. This iterative learning process is essential for optimizing the assistant's performance and ensuring its relevance in meeting the evolving needs of students and parents within the academic environment.

To further enhance chatbot performance, attention-based architectures within deep learning models have garnered attention. Attention mechanisms allow chatbots to focus on relevant parts of user input, improving contextual understanding and response accuracy. The transformer model, a groundbreaking architecture based entirely on attention, has shown superior performance in natural language processing (NLP) tasks compared to traditional sequential processing models like LSTMs.

Moreover, the Smart College Assistant stands out for its ability to deliver personalized and contextually relevant assistance at scale. Unlike traditional human-to-human support services that may experience delays and limitations during peak times, the assistant powered by Rasa can handle multiple inquiries simultaneously and provide 24/7 availability. This ensures that users receive immediate and accurate responses, leading to higher levels of client satisfaction and engagement.

The integration of Rasa's technology within the Smart College Assistant also underscores the importance of AI- driven solutions in enhancing educational support systems. By leveraging machine learning algorithms and natural language processing capabilities, the assistant not only streamlines information access but also contributes to a more intuitive and user-friendly academic experience. This innovative approach reflects a broader trend towards leveraging AI to optimize service delivery and improve overall efficiency within educational institutions.

In summary, the integration of Rasa's technology into the Smart College Assistant underscores the transformative potential of AI in enhancing educational support systems.

II. METHODOLOGY

This study adopts a research flow regarding chatbots with adjustments in several parts. There are 4 stages passed in this research, that are data collection, analysis and design, build the chatbot, implementation on the web, and testing.

A. Analysis and Design

The analysis and design stages are carried out before the chatbot creation stage with the aim of making it easier to create chatbots.

1) Requirement Analysis

Requirement analysis is carried out to dig up information and describe the data that will be used in making chatbots and the basic data is collected form the college website by scraping. In this process, data is grouped from the results of data collection that has been done previously. Data grouping is intended to simplify the process of entering data on intents, utter chatbot response templates, stories and actions int different files.

In making software, it is necessary to have a good design, so that the software that is built has good documentation to produce a system that has the needs that are in accordance with what the user wants.

B. Build The Chatbot

At this stage, a chatbot is made on a framework sense. This stage uses a chatbot workflow diagram.

1) Simulate Data

This step begins by simulating data based on general interaction dialogues and grouping each sentence into an intent topic, utter chatbot response template, and stories. Intent topics submitted as chatbot features must define the entities there.

2) Arrange Data to Stories

At this stage set the intent, utter and action on a story stream. The creation of stories will be placed on stories.yml.

3) Train NLU and Core

Data collection for training will be taken from the college website from which the data is scraped and they are grouped manually based on frequently asked questions by the parents an students about the college general queries and also for course related information. After all the data is ready, then train the data using the RASA Trainer.

4) Evaluate Model

Measure and evaluate the model made with precision score, recall score and F1-score. Evaluation of this model will be carried out in the terminal with the command provided by rasa. Then an assessment will be made on the intent whether the incoming questions will be categorized in the appropriate intent, so that later they will get the appropriate answer. This experiment will get the value of precision, recall, and the F1 score directly.

C. Implementation

This stage is carried out after the chatbot creation stage that is training the data and getting the model using rasa is complete. Implementation is done using Vue.js as frontend and google cloud platform and in turn using the url for integrating with website as a dropdown.

D. Testing

This stage analyzes the results of the implementation and testing of the chatbot application. Tests will be carried out on the model and functional testing of the chatbot using a Likert scale based. This Likert scale will have a scale of 4 by using questions in the form of a checklist. The calculation of this test is to calculate the average answer based on the scoring of each respondent's answer.

III. IMPLEMENTATION

A. Build The Chatbot

The following are the steps taken to create chatbot using the Rasa Framework.

1) Simulate Data

The data that has been obtained in data collection will be categorized into intent topics based on the frequently asked questions. In general, intent will be divided into greeting, general, conditional, selection, and random categories. The intent will be entered in the nlu.yml and domain.yml files. The nlu.yml intent will contain data for questions that will be trained later. Furthermore, based on the incoming data, it will be divided into several more detailed categories as shown in Table I.

TABLE I. Explanation of Intent Fuction

Intent	Function	Amount of data
greet	Categorize conversations as greetings	10
goodbye	Categorize conversations as goodbye	20
admission_open	Categorize conversations when admission details are asked	10
Inform_campus_life	Categorize conversations to give campus details	15
Inform_campus_discipline	Categorize conversations to give campus rules	20
Facilties_info	Categorize conversations about facilities	15
Hostel_info	Categorize conversations to give hostel information	10
Canteen_info	Categorize conversations to give canteen details	15
Hospital_info	Categorize conversations to give hospital details	10
Bank_info	Categorize conversations to give bank details	10
Placement_training_info	Categorize conversations to give placement training details	10
Placement_departmet_info	Categorize conversations to give department details	10
Placement_process	Categorize conversations to give placement process details	10
student_activity_clubs	Categorize conversations to give club details	12
Alumini_association	Categorize conversations to give alumni details	10
Alumni_testimonials	Categorize conversations to give testimonials about the college by alumni	10

To answer questions for each category of intent, an action is created as to determine the conversation action. This

action is defined in domain.yml with the name utter along with the intent. Each utter contains an answer template for intent questions contained in domain.yml. The list of actions on this chatbot is in Table II.

TABLE II. List of Utter

Number	Utter Name
1	utter_greet
2	utter_goodbye
3	utter_admissions_open
4	utter_achievements
5	utter_student_activity_clubs
6	utter_facilities_info
7	utter_hostel_info
8	utter_canteen_info
9	utter_health_service_info
10	utter_bank_info
11	utter_placement_info
12	utter_placement_process_info
13	utter_placement_training_info

2) Arrange Data to Stories

The next step is to create a conversation flow or stories. In stories.yml several conversations flow that might occur in this chatbot are created. The flow of the conversation will be replied with an answer from utter. In addition to preparing stories, test_stories are also prepared, which are similar to stories but filled with sentences that users might give to bots.

3) Train NLU and Core

Train or data training performed on the NLU, and Core of Rasa is affected by the configuration of the pipeline used. In this chatbot, the configuration is used as shown in Fig 1.

There are several components in the pipeline, the first of which are:

1. The tokenizer used in the chatbot pipeline is WhitespaceTokenizer. WhitespaceTokenizer will generate a token for each space-separated sequence of characters.
2. The feautizer in this pipeline is used by Regexfeaturizer, LexicalSyntacticFeaturizer and CountVectorsFeaturizer.

```
pipeline:
# # No configuration for the NLU pipeline was provided. The follow
# # If you'd like to customize it, uncomment and adjust the pipeli
# # See https://rasa.com/docs/rasa/tuning-your-model for more info
#   - name: WhitespaceTokenizer
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: rasa_nlu_examples.featurizers.dense.BytePairFeaturizer
    lang: id
    vs: 1000
    dim: 25
  - name: DIETClassifier
    epochs: 100
#     constrain_similarities: true
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
#     constrain_similarities: true
#   - name: FallbackClassifier
#     threshold: 0.3
#     ambiguity_threshold: 0.1

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
# # No configuration for policies was provided. The following defa
# # If you'd like to customize them, uncomment and adjust the poli
# # See https://rasa.com/docs/rasa/policies for more information.
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
#     constrain_similarities: true
  - name: RulePolicy
```

Fig 1. Pipeline chatbot

Some components on policies:

1. Memorization Policy
2. TED Policy

4) Evaluate Model

An evaluation of the model on the chatbot that has been made is carried out. Rasa itself facilitates users to carry out thorough validation and testing using test stories that have been created by users. Test stories are similar to stories in training data, but with messages from the user inserted into them. The command used to perform a performance test is to use "taste test".

Rasa test will test the trained model using the previously created test stories. The following below in Fig 2, Fig 3, and Fig 4 are the results of the evaluation using a sense test on the model.

Fig 2 is the intent confusion matrix table, the vertical axis shows the true table and the horizontal axis shows the predicted table. True table shows the original data that is in the training data and for the predicted table is the data predicted by the model. If the same true table and predicted table meet at the point where they intersect, it indicates that the model predicts the data according to the original data entered. Intent classification prediction works well, there is only two misclassifications when classifying intent.

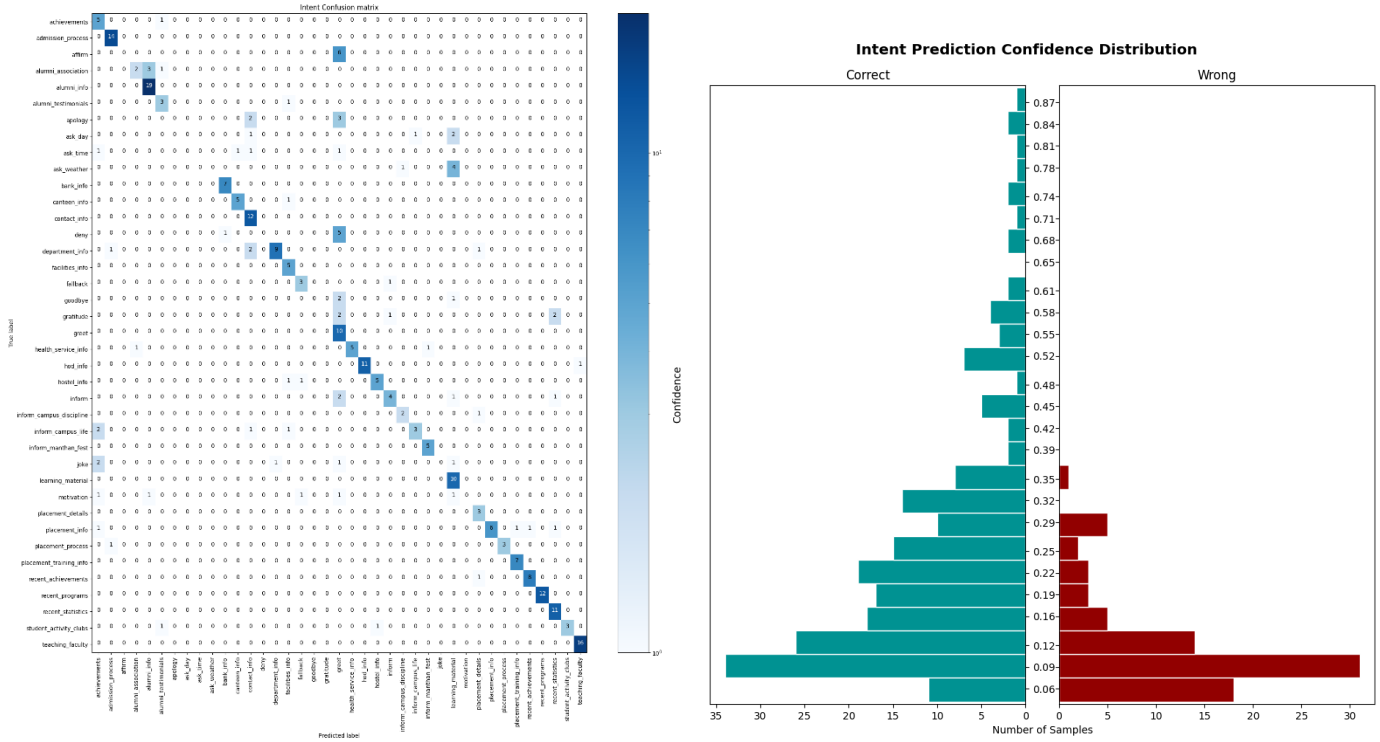

Fig 2. Intent Confusion Matrix

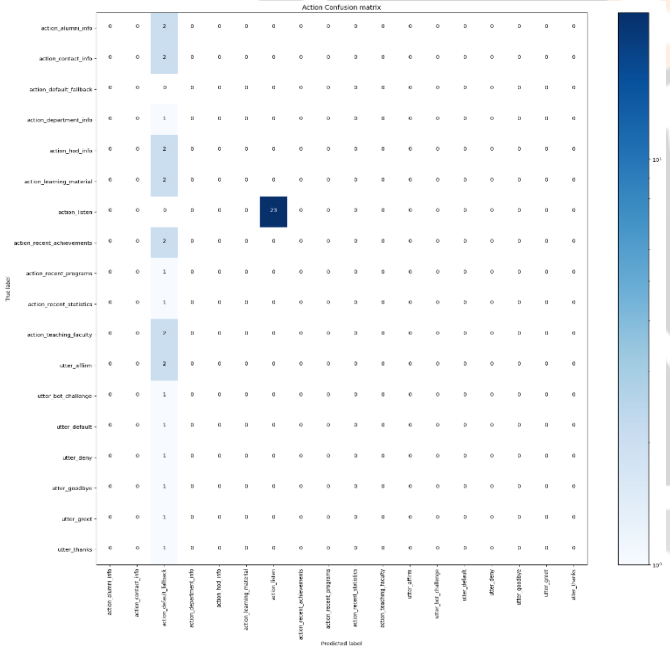

Fig 3. Story Confusion Matrix

Fig 3 is a story confusion matrix table. Similar to the confusion matrix intent, the difference is that it's not the intent that's being tested but from the story. The vertical axis shows the true table, and the horizontal axis shows the predicted table. For the story confusion matrix table, all the data tested are appropriate.

Fig 4. Intent Prediction Confidence Distribution

The following is an image of the intent prediction confidence distribution table. The numbers below indicate the number of samples held.

B. Implementation on The Web

The Rasa Chatbot is implemented using Vue.js as frontend. Google cloud is used because it is quite easy to use and can be used for free. After the deployment is complete, the URL of the project will be obtained Furthermore, this URL will be entered in the Chatbot-Widget template.

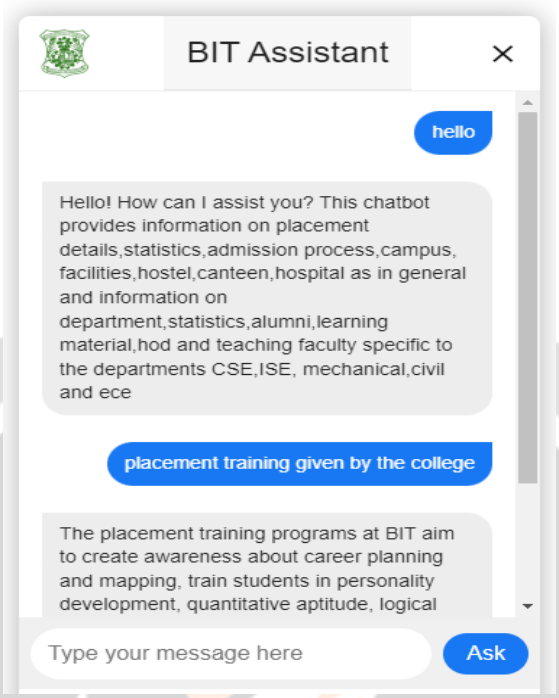

Fig 6. Chatbot Implementation Result

IV. TESTING

A functionality test on the chatbot was conducted on 10 people at the same time to find out whether the chatbot can function properly. A URL link is given for respondents so that respondents can test the chatbot in the form of a widget. A questionnaire is given which contains 8 questions with answer choices strongly agree, agree, quite agree and disagree. Each answer has a value, strongly agree is worth 4, agree is worth 3, quite agree is worth 2 and disagree is worth

1. From the responses that have been calculated based on the questionnaire that has been filled in and the total percentage result is 90.625%. The following in Fig 7 is a graph of the results of the functionality tests that have been carried out.

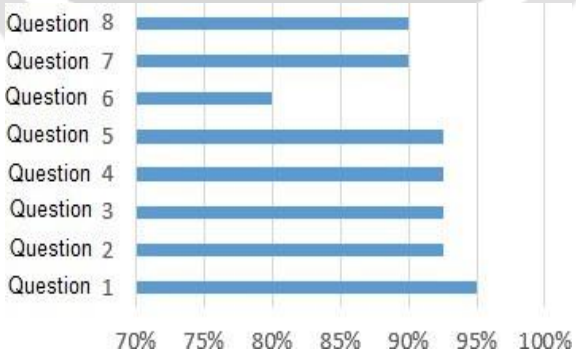

Fig 7. Chatbot Functionality Test Results Graph

Question 1 contains the question whether the chatbot appears on the page accessed by the respondent. There are suggestions from respondents regarding the welcome writing which is rather small so that it does not attract attention.

Question 2 contains the question whether the chatbot dialog box appears when the icon is pressed.

Question 3 contains the question whether the respondent can enter questions on the chatbot.

Question 4 contains the question whether the 'loading' sign appears when the chatbot is processing the question.

Question 5 contains questions the chatbot can provide answers after being given input.

· Question 6 contains questions Chatbot provides appropriate answers to questions given by respondents. Respondents several times got answers that were not in accordance with the questions given. The chatbot issues some default answers to a given question. This default answer is due to a fallback policy. In the Fallback policy, there is a confidence limit setting for an intent. If the incoming question is below the set confidence value, the default answer will appear instead of utter. In addition to setting the confidence value, respondents received answers that were not appropriate due to the limited topics from the chatbot which only focused on chatbot . Respondents gave suggestions to expand the scope of topics owned by the chatbot.

Question 7 contains the question of whether the use of chatbots is easily understood by respondents.

Question 8 contains the question of whether the chatbot is easy to use.

V. CONCLUSION

The college chatbot offers a simplified interface, consolidating query resolution for students and parents without the need for complex navigation. Employing language models and computational algorithms, the chatbot simulates human-like conversations, enabling natural language interactions. Users queries are efficiently handled through keyword analysis, swiftly retrieving relevant information from the database and referring trained models. The average value of accuracy is 0.94.

REFERENCES

[1] N. Deepika, M. M. Bala, and R. Kumar, "Design and implementation of intelligent virtual laboratory using RASA framework," Mater. Today Proc., Feb. 2021, doi: 10.1016/j.matpr.2021.01.226.

[2] L. Zhou, J. Gao, D. Li, and H.-Y. Shum, "The Design and Implementation of XiaoIce, an Empathetic Social Chatbot," Comput. Linguist., vol. 46, no. 1, pp. 53–93, Mar. 2020, doi: 10.1162/coli_a_00368.

[3] A. Tiha, "Intelligent Chatbot using Deep Learning Artist Identification from Artworks with Deep Learning [Python, Keras, CNN, Deep Learning] View project Big Data Application for Large Scale US Stock Market Data Analysis [JAVA, Apache Spark, Maven] View project," 2018, doi: 10.13140/RG.2.2.14006.75841.

[4] S. Roller et al., "Recipes for building an open-domain chatbot," arxiv.org, Apr. 2020.

[5] M. Chung, E. Ko, H. Joung, and S. J. Kim, "Chatbot e-service and customer satisfaction regarding luxury brands," J. Bus. Res., vol. 117, pp. 587–595, Sep. 2020, doi: 10.1016/j.jbusres.2018.10.004.

[6] Y. Windiatmoko, R. Rahmadi, and A. F. Hidayatullah, "Developing Facebook Chatbot Based on Deep Learning Using RASA Framework for University Enquiries," IOP Conf. Ser. Mater. Sci. Eng., vol. 1077, no. 1, p. 012060, Feb. 2021, doi: 10.1088/1757-899X/1077/1/012060.

[7] 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.

[8] U. Gnewuch, S. Morana, M. T. P. Adam, and A. Maedche, "Faster is not always better: Understanding the effect of dynamic response delays in human-chatbot interaction," 2018.

[9] Rakesh Kumar Sharma, "An Analytical Study and Review of open source Chatbot framework, Rasa," Int. J. Eng. Res., vol. V9, no. 06, Jun. 2020, doi: 10.17577/IJERTV9IS060723.

[10] A. Singh, K. Ramasubramanian, and S. Shivam, "Introduction to Microsoft Bot, RASA, and Google Dialogflow," in Building an Enterprise Chatbot, Berkeley, CA: Apress, 2019, pp. 281–302.

[11] S. Meshram, N. Naik, V. Megha, … T. M.-… A. C. on, and U. 2021, "College Enquiry Chatbot using Rasa Framework," ieeexplore.ieee.org.