INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 3 - V7I3-2193)
Available online at: https://www.ijariit.com

University chatbot with database integration using RASA

Harith L. K.
harithlk316@gmail.com
R. V. College of Engineering, Bengaluru, Karnataka

K. Vadhi Raja
vrpk07@gmail.com
R. V. College of Engineering, Bengaluru, Karnataka

Dr. G. S. Mamatha
mamathags@rvce.edu.in
R. V. College of Engineering, Bengaluru, Karnataka

ABSTRACT

Chatbot is an Artificial Intelligence (AI) software that simulates and processes human conversations as if the humans were having a real conversation with another human. They are used in various business applications to help address human queries and assist them with important details. RASA is an open-source machine learning framework for building chatbots. RASA leverages Natural Language Understanding (NLU) to break down user messages into intent and entities which help form an appropriate response for the user's messages. The RASA conversational assistant has several components that help predict the responses and also tune the model to get better results from the model. In this paper, this paper will provide the design of a chatbot that handles basic queries of a particular university that could be used on the university website. This paper also explores how individual components of the RASA pipeline help curate the response for the users.

Keywords: *RASA, Natural Language Understanding (NLU), Chatbot,*

1. INTRODUCTION

Automation of tasks has been growing rapidly due to an increase in technological resources and tools. One such area is the automation of customer support. Many of the companies now have automated responsive chatbots to provide extended customer service. The main intention of using chatbots is to provide support throughout the day 24/7. They are highly accessible, customizable, and provide instant responses to the user. They can be customized to perform actions based on the inputs given to it [1]. In this paper, Rasa is discussed, an open-source conversational framework to build a university chatbot. Rasa consists of various tools like Rasa NLU, Rasa Core which are open-source python libraries to provide distributive functionality handling systematically. Natural language understanding (NLU) is the most important aspect of a conversational framework. Rasa NLU enables the application to tokenize and define the user input and respond accordingly [2].

A university chatbot is an application that can provide an immediate response to Students/Parents and even the Staff wherever necessary. Queries like Admission details, Fee structure, College/Department details, Placement details are handled by the bot and respective responses will be given to the user.

2. RASA

Rasa is an open-source framework based on machine learning to automate textual and voice-based assistants. RASA has two core modules, RASA NLU and RASA Core [3]. RASA NLU is RASA's Natural Language Understanding module which uses machine learning and Natural Language Processing (NLP) to extract the important details which are contained within a message such as intents and entities [4]. Intents refer to the goal or the purpose with which the user sends the message to a chatbot [5]. Entities are the useful data that are present within the user's message which decides how the dialog flow should move further. These entities add value to the user's goal of communication with the chatbot [5]. RASA allows us to handle intents and entities more simply. The intents are mentioned in a yaml file, where all the user intents along with their example messages would satisfy the respective intent. The entities can also be provided along with the intent examples by placing them within square brackets and mentioning the name of the intent inside brackets after the entity. Ex: [Computer Science](department). Here, Computer Science is the entity belonging to an entity named department.

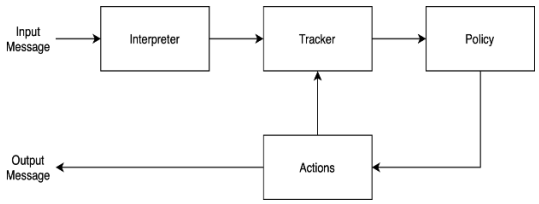

Fig 1. RASA Architecture

The architecture above shows a simple workflow of how the data entered by the user will go through a number of different stages before responding. The interpreter block is RASA's Natural Language Understanding unit which breaks down the input into a set of intents and entities which help the dialog flow module or the core module to predict an appropriate message for the response [6]. Tracker helps us record all the data and store it for the purpose of using it in further stages of the workflow. Tracker helps us to maintain the state of the conversation [6].

Policies help identify the next best action to be taken [6] based on the input provided by the intent and entities obtained from the interpreter module and the state information from the tracker. The respective action triggered will give out a response to the user to the message he or she sent. The action triggered is logged into the tracker [6].

3. SYSTEM DESIGN

This section describes the proposed system along with some of the features offered by the chatbot. RASA's NLU and Core modules are leveraged to offer the features for the proposed system. The connection of RASA chatbot customized version of the chatroom frontend application which provides with the User Interface for our conversational assistant.

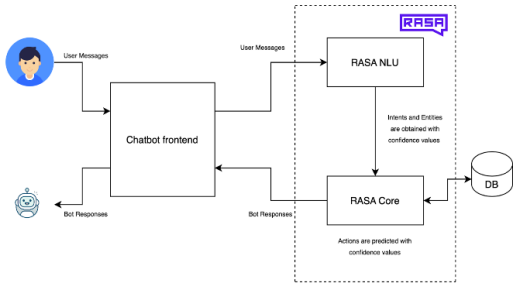

Figure 2. Architecture for the university chatbot application

A high-level overview of the University Chatbot application is shown in Figure 2. As you can see, there is a clear distinction between the front end and the RASA server. The RASA server serves up the bot responses based on the user's messages using the RASA API endpoint. The message transfer happens in JSON format. The users are presented with a front-end application where the users can post their messages, ask queries, and have conversations with the assistant. These messages sent to this front-end application is passed on to the RASA NLU module, where the intents and entities of the messages are captured. The intents and entities are then used by the core module of RASA to obtain a response for the respective message. There are files written using YAML to set examples for the intents used and also to capture the entities. YAML files are used to write the responses which are sent back to the user based on the core's prediction. The NLU module predicts the intents and entities with a certain confidence level. The intent with the highest confidence level is used by the core module to predict the response. RASA has the concept of custom actions which provides us the ability to write custom code which can be leveraged to give out responses based on queries to a database or APIs.

3.1 Database Connection

The architecture of the chat application shows the connection with database by the RASA Core module in Figure 2. In the given system RASA server is connected with a SQLite database. This database contains details regarding the university which will help to answer queries made by the users.

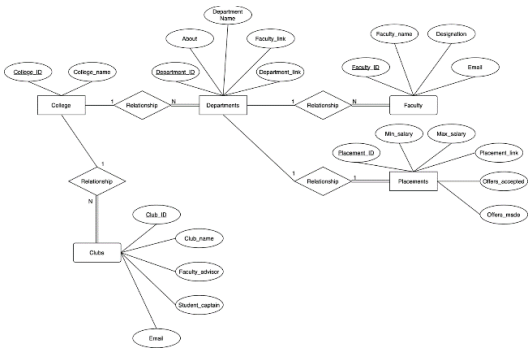

Figure 3. Entity-Relationship Diagram for the University Database

The Entity-Relationship diagram for the SQLite database is as shown in Figure 3. The database has several entities. The College entity refers to the college database which identifies the college. The departments entity will hold all the details relating to a department, which will help to answer queries related to the department such as about department, departments list, and others. The Faculty Entity will hold the data related to faculties and will help to fetch the faculty information when queried. Placements Entity helps to obtain placement information for individual departments. Clubs' entity which is directly related to the college entity will help to fetch the details about all the clubs present at the college. The database connection has to be made from the actions module in RASA which helps to build our custom actions in RASA to serve out responses for the users.

Our system uses the database to query for details such as department details, placements of individual departments, getting details of clubs at the college, and other details. The core module runs the custom code written for querying the database for the above-mentioned details to fetch the appropriate responses for our chatbot to respond to the user.

3.2 NLU Pipeline

RASA NLU is modularized into pipelines [3]. The pipelines the processing stages which the user messages will go through before they are classified as to be of a specific intent having one or more entities which in turn helps to predict the bot message. The components specified in the pipeline are executed one after the other in which the output of one component of the pipeline is used by the subsequent components in the pipeline. In the proposed system, two different pipelines were tried out, pre-trained embedding and supervised embeddings with DIET classifier and compare the performance of the chatbot for both cases. The pre-trained embeddings come with pre-trained word embeddings [7], which helps to obtain similarities between the words which have not been used in the training data. This model helps to match with other words related to the domain which is helpful in case the training examples provided are less in number. The supervised embedding pipeline used in the proposed system uses the DIET (Dual Intent and Entity Transformer) classifier which is used for simultaneous classification of intent and identification of entities present in the user's message. The three most common and important steps which any pipeline does are the tokenization of the words in the message, feature extraction from the tokens, and classification of the intents and entities present in the message. Tokenization involves separating the words in the message which are then submitted to the featurizer to obtain features of the message based on individual words present in the message. Feature extraction uses the words from the tokenizer and then calculates meaningful numbers based on the significance of the words in the message. These numbers are then passed to the classifier, which helps to classify the intents and also in identifying the entities. The classification of intents and identification of the entities then help serve the purpose of selecting a response by the Core module of RASA in assistance with the Database.

3.3 DIET Classifier

Dual Intent and Entity Transformer, also shortly known as DIET classifier, is a state-of-the-art classifier which is lightweight and is designed with a multitask transformer architecture for NLU [8]. It handles both the functionalities i.e., Intent classification and Entity recognition both together. It also provides the ability to plug and play various models which are embedded and already pre-trained with BERT, ConveRT, GloVe etc.

- It is built as a modular architecture which makes it highly feasible to be incorporated into a software development workflow.
- It is in-line with the existing large scale pre-trained models in terms of performance and accuracy.
- Contains wide functionalities to perform complex predictions.

DIET classification uses a sequence model which considers the order of the word, thus improving the performance [8].

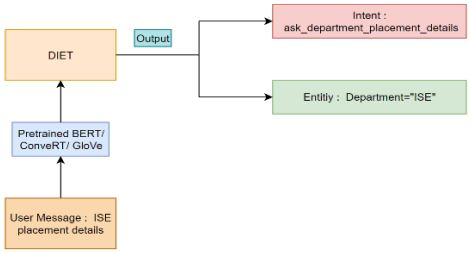

Figure 4: Illustration showing the working of DIET classifier.

4. IMPLEMENTATION

The primary step in developing the required application is to have a trained Rasa conversational AI model which can then be integrated with the required application, this might include a website, a simple UI to get a clear interface to communicate with the chatbot. In order to train the required model, a Rasa project needs to be initialised using "rasa init" command. The NLU component of the rasa tool takes care of identifying the message or query input by the user.

After the project is initialized, Rasa creates distributive scripts and folders which handle the functionalities like:

- nlu.yml: Define the intents and the message or queries that the intents can be associated with.
- rules.yml: Define the default rules and actions to be performed in case of the said rules.
- stories.yml: Define the series of steps and intents to be performed for a particular "story".
- Actions.py: A python script defined to connect to the database and retrieve the relevant information of a particular intent. The functions to retrieve the required information can be defined for each intent can be defined in the script. Rephrasing the user input to match the intents can also be done to make information retrieval and intent classification unchallenging.
- Test_strories.yml: Contains the test file with intents and actions to check whether the chatbot is performing as it should.
- Config.yml: Define the feature filters such as WhiteSpaceTokenizer – used to read input as tokens without whitespaces, tabs and newlines, RegexFeaturizer – To create a list of regular expressions which are in the data during the training of the model.
- Credentials.yml: Define the socket IO connection for each session, and also specify the port on which the rasa server runs.
- Domain.yml: Define the responses for each intent specified in the NLU, these responses should be given based on the intent detected by the intent classifier. The functions defined in the python script are also called here in the form of "actions".
- Endpoints.yml: To define the end point URL for the frontend or any other API-based application to request and connect to the rasa server model.

5. RESULTS AND EVALUATION

This section provides results obtained from the University chatbot. The results include intent classification confidence for two different pipelines and the confusion matrix which describes how well the model predicted the respective intents. This will help to evaluate the model's performance and compare with the pipelines.

5.1 Intent Classification for Pre-Trained Embeddings Pipeline

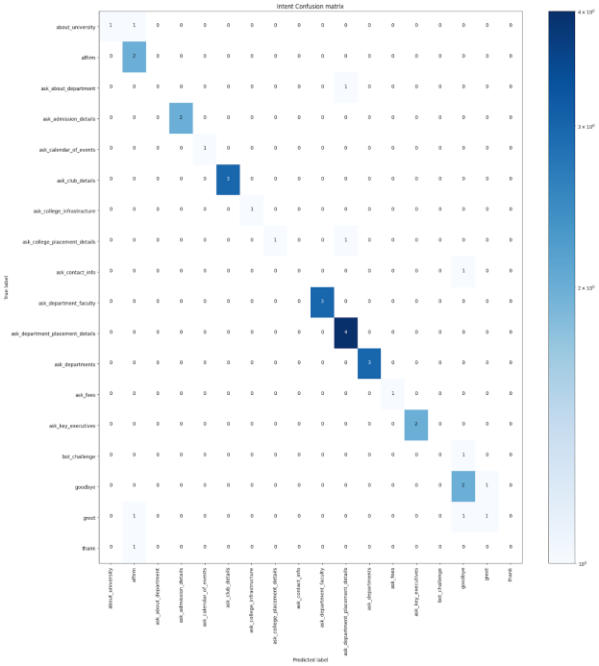

Figure 5. Intent classification for Pretrained Embedding Pipeline

Figure 5 shows the confusion matrix for the intent classification using Pretrained embeddings pipeline. It can be seen that several intents have been mis predicted. The intents related to university queries have mostly been predicted correctly.

Figure 6 shows the intent classification confidence distribution which shows the confidence levels at which different entities have been predicted. The left side of Figure 5 shows the confidence levels at which the intents were correctly predicted. The pretrained embeddings model has many intents predicted correctly with lesser confidence as shown in the Figure 5. This shows that the model has high susceptibility to predict the wrong intent. The Figure 5 also shows some red lines on the right side which represents the confidence levels at which the wrong intents were predicted. The pipeline predicts the wrong intents with less confidence, which is a good optimization for the model.

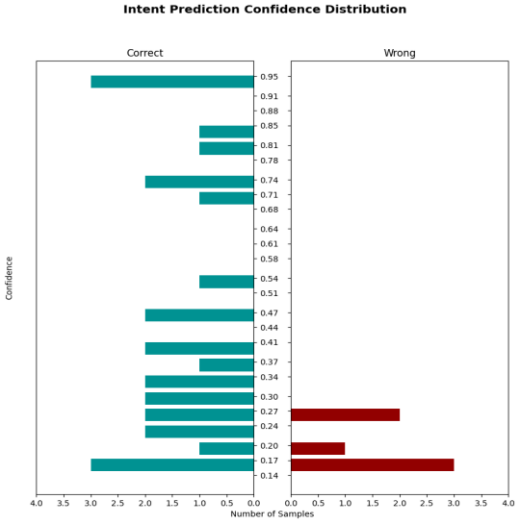

Figure 6: Intent Prediction Confidence Distribution for Pretrained Embeddings Pipeline

5.2 Intent Classification for Supervised Embeddings Pipeline with Diet Classifier

The intent confusion matrix shown in the Figure 6 shows better results for predicting the respective intent by the model when ran on a test sample data. Even though the training data is less, good results can be seen for intent prediction. The Figure 7. shows the confidence levels for the intent prediction. As it can be seen that most of the values on the left side fall on the top part of the graph, which shows that the model predicts the correct intents with high confidence and the red line on the right side indicates that the model predicted only a single intent wrong with moderate confidence. This shows that the DIET classifier with the Supervised Embedding Pipeline [9] performs better than the Pretrained Embeddings Pipeline for the training and testing samples available.

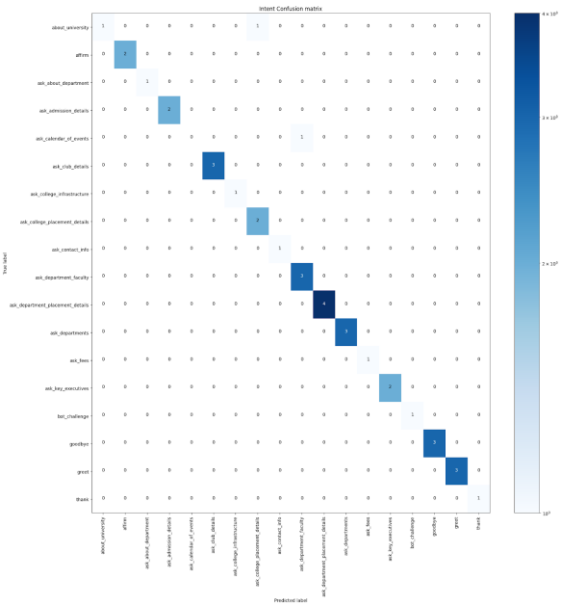

Figure 7: Intent Confusion Matrix for Supervised Embedding Pipeline

Figure 8: Intent Prediction Confidence Distribution for Supervised Embedding Pipeline

6. CONCLUSION

The university query bot classifies the intent based on the input given using the DIET classifier. The python script connects the application to the database and enables communication between the application and the database. The main functionality required for an appropriate and accurate response in intent classification, which is a major part of Natural Language Understanding (NLU) and the result obtained from this is the final prediction or interpretation of the input query. We could see that the DIET Classifier (Dual Intent and Entity Transformer) with Supervised Embedding Pipeline [9] showed better results than the Pretrained Embedding Pipeline. The reason for this is because in DIET classification, the transformation architecture handles the Intent recognition functionality and the recognition of the respective entity together. The DIET classifier total loss is trained and determined by the total for entity loss, mask loss and intent loss.

Optimization of the algorithms has been and always will be a work for the future. The efficiency of the algorithms determines the extensive use of it. In Artificial Intelligence, there is always a scope for improvisation by performing certain trade-offs and finding the point of utmost efficiency. DIET Classification might be the state-of-the-art classifier for Rasa, but it has its limitations. Due to increase in demand of such automated conversation chatbots, there is an increased requirement to develop an algorithm that can handle large scale data processing very efficiently. Thus, it can be stated that there is much scope for such development since the AI market [10] is always on demand by the institutions to automate their management style for enquiries.

7. REFERENCES

[1]. Andrew Rafla and Casey Kennington. 2019. “Incrementalizing rasa’s open-source natural language understanding pipeline.” *arXiv preprint arXiv:1907.05403*.

[2]. A. Abdellatif, K. Badran, D. Costa and E. Shihab, "A Comparison of Natural Language Understanding Platforms for Chatbots in Software Engineering," in *IEEE Transactions on Software Engineering*, doi: 10.1109/TSE.2021.3078384.

[3]. Mohit Jain, Ramachandra Kota, Pratyush Kumar, and Shwetak N. Patel. 2018. “Convey: Exploring the Use of a Context View for Chatbots.” In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). Association for Computing Machinery, New York, NY, USA, Paper 468, 1–6. DOI: https://doi.org/10.1145/3173574.3174042

[4]. R. Sharma, "An analytical study and review of open source chatbot framework rasa", *International Journal of Engineering Research and*, vol. V9, no. 06, 2020.

[5]. A. Stoica, T. Kadar, C. Lemnaru, R. Potolea and M. Dînşoreanu, "The Impact of Data Challenges on Intent Detection and Slot Filling for the Home Assistant Scenario," 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP), 2019, pp. 41-47, doi: 10.1109/ICCP48234.2019.8959642.

[6]. 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," 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), 2021, pp. 25-29, doi: 10.1109/EIConCIT50028.2021.9431913.

[7]. Vladimir Vlasov, Johannes E. M. Mosig, and Alan Nichol “*Dialogue Transformers*” 2019, arXiv Cornell University, Submitted on 1 Oct 2019 ,last revised 1 May 2020.

[8]. Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, Alan Nichol *“DIET: Lightweight Language Understanding for Dialogue Systems”* Arxiv: archive 2004.09936

[9]. Mihir Kale, Aditya Siddhant, Sreyashi Nag, Radhika Parik, Matthias Grabmair & Anthony Tomasic “*Supervised Contextual Embeddings for Transfer Learning In Natural Language Processing Tasks*” Language Technologies Institute Carnegie Mellon University Pittsburgh, PA 15213, USA

[10]. Yang Cheng, Hua Jiang “*Customer–brand relationship in the era of artificial intelligence: understanding the role of chatbot marketing efforts*”. Department of Communication, North Carolina State University, Raleigh, North Carolina, USA - Journal of Product & Brand Management © Emerald Publishing Limited [ISSN 1061-0421] [DOI 10.1108/JPBM-05-2020-2907]

