NATURAL LANGUAGE CHATBOT

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Surbhi Sood (201111)

Under the guidance & supervision of

Mr. Arvind Kumar

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled "Natural Language Chatbot" in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "**Surbhi Sood, 201111** " during the period from August 2023 to May 2024 under the supervision of **Mr. Arvind Kumar**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Surbhi Sood
(201111)

The above statement made is correct to the best of my knowledge.

(Mr. Arvind Kumar)
Assistant Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

DECLARATION

We hereby declare that the work presented in this report entitled ‘Natural Language Chatbot’ in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering / Information Technology submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of Arvind Kumar (Assistant Professor, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Surbhi Sood

Roll No.: 201111

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Arvind Kumar

Designation: Assistant Professor

Department: CSE & IT

Dated:

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing making it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Mr. Arvind Kumar , Assistant Professor (Grade II)** , Department of CSE & IT Jaypee University of Information Technology, Wakhnaghat . Deep Knowledge & keen interest of my supervisor in the field of "**Machine Learning and Artificial Intelligence**" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible for partial completion of this project.

I would also generously thank each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and no instructing, which have developed their convenient help and facilitated my undertaking.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The aim of the project is to change customer interactions by introducing RASA based chatbots designed specifically for the banking industry. The chatbot uses RASA's natural language understanding (NLU) to understand the user's intent and make the interaction more efficient. The confidentiality of financial transactions is proven by these standards, leading to security and social security. Chatbots can manage conversations, answering complex questions and making it easier for customers. The company's business knowledge can be incorporated through customization, which also increases the effectiveness of chatbots in business support and financial investigations. The chatbot has gone through a rigorous training process and is constantly evaluated to improve its effectiveness and accuracy. Real-world applications demonstrate the effectiveness of chatbots in improving customer experience, increasing response time, and creating more banking interactions. This initiative represents a significant step forward in customer service through the deployment of advanced RASA chatbots in banking.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The artificial intelligence-supported system, called "natural language chatbot", can communicate with users like a human. Chatbots use machine learning and natural language processing (NLP) to understand human intent, relationships, and context. This allows users to read their questions and offer answers in a way that makes the conversation real. Natural language chatbots provide diverse, context-aware interactions using algorithms and models that learn and change over time. This makes them useful in many applications such as customer service, data storage, and business automation.

The project uses an open source called **Rasa** to create an artificial intelligence-based chatbot. Rasa is a Python framework and toolset for creating, building, and deploying intelligent chatbots that can understand and react to natural language. Rasa stands out by emphasizing transparency and flexibility by giving developers more power to customize chatbot behavior. It has two main components: Rasa Core for speech control and Rasa NLU for language comprehension. Rasa Core makes it easy to create interactive, context-aware conversations, while Rasa NLU allows chatbots to understand the user's intent, context, and context. Rasa is an open source that facilitates design and community collaboration, making it a popular choice for creating powerful and flexible AI interactions.

Our Natural Language Chatbot, strengthened by the powerful RASA framework, is a sophisticated solution that is set to revolutionize user interactions in the rapidly changing field of conversational AI. Our chatbot uses sophisticated natural language processing to understand user intents, entities, and context, going beyond conventional rule-based systems. Built upon the adaptable framework of open-source RASA, our project provides a canvas for easy extension and customization, adapting to the unique requirements of various applications.

1.2 PROBLEM STATEMENT

Creating a Rasa bot capable of understanding and responding to user queries in natural language, with a focus on providing accurate and context-aware conversational experiences. Below are some key points which were they key issues which were taken under consideration before creation of this project:

1. Inefficiencies in Traditional Support Systems:
   - Current customer support systems often face challenges in providing timely and personalized assistance.
   - Delays, lack of contextual understanding, and a uniform response structure are common issues.
2. Adaptive Learning and Continuous Improvement:
   - With the help of RASA, the chatbot will learn and adjust over time, going beyond static responses.
   - Machine learning will guarantee that the chatbot keeps becoming better and better at managing intricate client interactions.
3. Streamlining Support Processes:
   - The project aims to improve customer satisfaction by reducing response times, streamlining processes, and incorporating RASA into customer support.
   - The objective is to offer a support system that is more effective and efficient and that adapts to the changing needs of users.

1.3 OBJECTIVES

The objectives of a Rasa chatbot can be summarized in the following key points:

- Enhanced Customer Support
- Information Retrieval
- Task Automation
- Contextual Conversations
- Continuous Improvement
- Natural Language Understanding

In brief, the primary objective of a Rasa chatbot is to enhance user engagement, streamline support, automate tasks, and provide a seamless and secure conversational experience.

1.4 SIGNIFICANCE

Significance of the Natural Language Chatbot Project:

1. Enhanced Customer Experience:

   The addition of a chatbot driven by RASA demonstrates a dedication to improving customer satisfaction. Within the customer support ecosystem, the project aims to improve overall user satisfaction by enabling more natural and context-aware interactions.

2. Efficiency in Support Processes:

   The RASA chatbot project, which denotes a break from conventional support systems, is positioned to streamline support procedures. By expediting responses and decreasing latency, the chatbot enhances resource optimization and enhances service provision.

3. Customization and Flexibility:

   The fact that RASA is open-source adds significance by enabling a high degree of flexibility and customization. This guarantees that the chatbot can be customized to meet the particular demands of a project and take into account the special needs of the customer support setting.

Motivation behind the RASA Chatbot Project:

1. Efficient and Timely Support:

   The aim is to offer users faster, more effective support. The potential for revolutionizing response times and improving the overall effectiveness of customer support interactions is presented by the capabilities of RASA.

2. Adaptive Learning for User-Centric Solutions:

   The goal is to enhance the chatbots capacity for learning and adaptation, so that it can respond in a way that is both relevant and personalized, catering to each user's particular needs.

3. Striving for Technological Excellence

   The driving force is a dedication to technological superiority. We hope to not only meet industry standards by implementing RASA into the project, but also to create new standards for creative and efficient customer support solutions

1.5 ORGANIZATION OF PROJECT REPORT

In this project reports we have included various aspects of the project in the form of chapters, each chapter has some specific contents which can be briefed as follows:

- CHAPTER 2: In this chapter we have covered a thorough description of 8 research papers in which we have mentioned its tools and technologies , results and key gaps found in the research papers related to RASA and natural language chatbots.

- CHAPTER 3: This chapter covers the system development of our project and we have covered all the requirement , analysis , implementations and key challenges faced during the development of this project.

- CHAPTER 4: This chapter covers the testing strategies employed for the model testing of RASA bot , we have covered the test case instances on which the test bot was tested and have mentioned the outcomes obtained by theses test cases.

- CHAPTER 5: This Chapter covers the evaluation of results from the working model of our chatbot we have attached several snippets showcasing the working model of our chatbot under different scenarios.

- CHAPTER 6: This chapter contains the conclusion and future scope for the RASA bot, in this we have covered the different aspects for which we can use RASA and how it is having its own pros and cons.

This report will give you a good insight about RASA and Natural Language Understanding. With the idea of how, when, where and why RASA is a great tool and better than conventional bots.

CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

2.1.1 Chatbot using RASA [1]

Abstract:

In this study, we explore the evolving landscape of chatbot technology, focusing on RASA—an open-source framework recognized for its advanced Natural Language Understanding (NLU) capabilities and the powerful Dual Intent and Entity Transformer (DIET) model. RASA has transcended traditional chatbot roles, demonstrating proficiency in complex tasks and human-like interactions. The study delves into RASA Core's multifaceted features, emphasizing its adaptability for real-world applications, database and API interactions, and interactive learning through reinforcement techniques. The research evaluates RASA's functionality in the PyCharm Integrated Development Environment (IDE) on Windows, providing insights into its robustness and potential for creating highly capable conversational AI assistants.

Tools and Technologies:

1. Rasa Framework: Open-source machine learning framework for AI chatbots and Comprises Rasa NLU for user message understanding and Rasa Core for conversation management.

2. Natural Language Processing (NLP): Techniques employed to enable the chatbot to understand and process natural language input from users and Facilitates intent classification and entity extraction for improved interaction

3. API Integration: Chatbot integrates with external APIs like OpenWeatherMap and Polygon.io and Enhances functionality by fetching real-time data, such as weather information and stock prices.

4. Python: Programming language used for developing custom actions and implementing various chatbot functionalities and Enables interactions with users and data retrieval from external APIs.

Results:

1. Weather Information: Chatbot provides real-time weather information for specified locations. Users can inquire about weather conditions using natural language queries.

2. Stock Price Data: Users request and receive real-time stock price data for specific stock ticker symbols and Integration with Polygon.io API keeps users updated on stock market trends.

3. Intent Recognition: Chatbot effectively recognizes user intents and responds accordingly and Demonstrates an understanding of the context of user queries and prompts for missing information when necessary.

4. Conversational Flow: Rasa Core's probabilistic model enables smooth, context-aware conversations. Follows predefined stories and actions, contributing to natural and dynamic interactions

2.1.2 Implementation of an Educational Chatbot using RASA Framework [2]

Abstract:

The rapid growth of Artificial Intelligence (AI), Big Data, and Internet-of-Things (IoT) technologies has led to the widespread application of chatbots across diverse domains. These AI-powered virtual agents are now prominently

utilised in sectors such as e-commerce, banking, healthcare, and customer service. However, there remains an untapped potential for deploying chatbots in the educational landscape, particularly to engage and motivate rural students who often lack access to advanced educational resources. In response to this need, this study endeavours to create an interactive and personalised educational web-based chatbot using the Rasa framework, aimed at transforming the way rural students interact with educational content.

Tools and Technology:

1. Rasa Framework: The primary technology used for building the chatbot. Rasa is an open-source framework for building conversational AI.

2. Flask: Flask is a micro web framework used for developing the web application that hosts the chatbot.

3. Firebase: Firebase is used for data storage. Firebase is a cloud-based platform that provides various services, including real-time database and authentication.

4. Content-Based Filtering: Content-based filtering is used for providing personalized recommendations to users based on their quiz scores and learning preferences.

Results:

The paper does not provide specific quantitative results but mentions the following
outcomes and functionalities:

1. Course Learning: Users can choose from a list of courses in subjects like Maths, Science, Computer Science, and Aptitude.

2. Quizzes: Users can take quizzes related to the courses they've studied. The system tracks their quiz scores.

3. Recommendation System: The chatbot uses content-based filtering to recommend courses to users based on their quiz performance. Faculty Information: Users can find information about faculties and even set appointments with them for doubt clarification.

4. Chatbot Interaction: The chatbot is designed to interact with users, answer their queries, and guide them through the learning process.

2.1.3 RASA Chatbot Using AI [3]

Abstract:

The last decade has witnessed remarkable technological advancements driven by Artificial Intelligence (AI), Big Data, Internet of Things (IoT), and other innovative technologies. Among these advancements, chatbots have emerged as valuable tools, simulating human conversations and automating mundane tasks. This study focuses on creating a web-based chatbot called "College Enquiry Chatbot" for the education sector, built using the Rasa technology. This open-source platform employs Rasa Core and Rasa Natural Language Understanding (NLU) to construct a contextual AI chatbot. NLU is responsible for understanding user intent and extracting essential information, while Rasa Core employs a Recurrent Neural Network (RNN) to generate responses. The chatbot's performance is evaluated using metrics like Precision, Accuracy, and F1 Score, resulting in average scores of 0.628, 0.725, and 0.669, respectively. The chatbot's accuracy, accessibility, and low maintenance make it a promising tool for various applications beyond education, streamlining inquiry processes.

Tools and Technology:

1 Rasa Framework: The primary technology used for building the College Enquiry Chatbot. Rasa is an open-source framework for building conversational AI.

2 Flask: Flask is likely used for developing the web-based application that hosts the chatbot.

3 Rasa NLU (Natural Language Understanding): Rasa NLU is part of the Rasa framework and is used for understanding and extracting intent and entities from user input.

4 Conditional Random Fields: Mentioned as a part of the featurization process, Conditional Random Fields are used for predicting sequences and can be employed for entity extraction

Results:

The paper provides some results and findings in the implementation part of the College Enquiry Chatbot:

1. Performance Measures: The paper mentioned that the performance measures like Precision, Accuracy, and F1 Score, which have average values of 0.628, 0.725, and 0.669, respectively. These measures represent that the model can understand and respond to user queries efficiently.

2. Architecture: The paper shows the system architecture, data flow diagrams, and flowcharts, providing a better way of understanding of how the chatbot is created and works.

3. API Communication: The chatbot appears to have the capability to communicate with APIs, which could enable it to fetch data or perform actions like booking tickets.

4. Mathematical Model: The paper describes the use of mathematical models like Bag of Words (BoW) and Conditional Random Fields for various tasks, such as feature extraction and entity recognition.

2.1.4 College Enquiry CHATBOT using RASA [4]

Abstract:

The paper discusses the development of a College Enquiry Chatbot using the RASA open-source framework. With the proliferation of technology and smartphones, chatbots have become a vital part of the digital landscape, providing instant responses to user queries. The College Enquiry Chatbot is designed to efficiently The chatbot's underlying framework, RASA, is comprised of two essential components: Rasa NLU and Rasa Core. Rasa Core plays a crucial role in managing the chatbot's conversational flow and enables the creation of more sophisticated and customized chatbots. Rasa NLU, on the other hand, equips developers with the tools and technology needed to understand and interpret user input, including determining intent and extracting entities. The implementation of the College Enquiry Chatbot necessitates specific hardware and software requirements, including an i3 processor-based computer with 2GB of RAM, Rasa, and Python 3.6 or higher. The primary objective of this project is to develop a chatbot capable of efficiently addressing student inquiries, searching for relevant information, and providing solutions.

Tools and Technologies:

1. Rasa: The chatbot is implemented using Rasa, which is an open-source framework for building AI chatbots. Rasa consists of two components: Rasa NLU (Natural Language Understanding) and Rasa Core.

2. Python 3.6 or higher: Python is used as the programming language for developing the chatbot.

3. Machine Learning: Machine learning techniques are employed for tasks such as intent classification and entity extraction.

4. Bag of Words (BoW): BoW is used for feature extraction from text data. It involves the collection of data, designing a vocabulary, and creating document vectors.

5. Conditional Random Fields (CRF): CRF is a discriminative model used for predicting sequences. It helps in improving the model's prediction by considering contextual information from previous labels.

Results:

The paper presents some results and screenshots of the chatbot's functionalities:

1. Front Page of Chatbot: The chatbot provides a user-friendly interface with options for the user to select their query.

2. About Us Option: This option provides details about the college, including NAAC grade, and links to the official social media pages.

3. Contact Us Option: Users can find contact information for the college, including the website and WhatsApp number.

4. MHT-CET Mock Paper Option: This option allows users to take a mock test for better practice.

5. Courses Option: Users can view the available BE and ME courses

2.1.5 Predicting Frequently Asked Questions (FAQs) on the COVID-19 Chatbot using the DIET Classifier [5]

Abstract:

A popular dialogue system in the field of natural language processing (NLP) is the chatbot. Chatbots aim to create conversations between humans and machines. COVID-19 is a member of the Coronavidae (CoV) family of the Corona viirinae family which causes the respiratory system to become severe in humans. This paper predicts chatbot answers to questions about COVID-19 with the RASA framework and uses the DIET Classifier pipeline for 300 training data. The test results with the DIET Classifier model on rasa.core.test and rasa.nlu.test provided confidence values of F1-Score, precision, and accuracy for the correct answer to the question about COVID-19, namely 1.0 with a percentage of around 85%.

Tools & Technologies:

1. DIET Classifier - DIET (Dual Intent and Entity Transformer) Classifier is a state-of-the-art natural language processing model designed for intent and entity recognition tasks in dialogue systems, employing a transformer-based architecture.

2. RASA - The primary technology used for building the chatbot. Rasa is an open-source framework for building conversational AI.

3. NLP - Natural Language Processing (NLP) is a field of artificial intelligence that involves the interaction between computers and human language, enabling machines to understand, interpret, and generate human-like text.

Results:

1. The DIET Classifier model in the RASA framework predicts answers on the chatbots related to COVID-19 information which uses around 300 epochs without normalizing data.

2. RASA test will show the results of the DIET Classifier model on rasa.core.test
   and rasa.nlu.test by evaluating 7 stories and the results in END-TO-END level namely Correct, F1-Score, Precision, Accuracy, In-data fraction, Confusion Matrix without normalization on rasa core test

3. The effectiveness of the DIET model pre-training with the use of embeddings
   always provides the best result among different data sets.

4. In the future development of the COVID-19 chatbot, other models such as BERT, GloVe are recommended as a comparison of the prediction results obtained for the DIET Classifier model.

2.1.6 Jollity Chatbot- A contextual AI Assistant [6]

Abstract:

Chatbot is a software application that can stimulate a conversation via text, instead of direct contact with a live human through messaging applications, websites and mobile applications. Chatbot applications help to make interactions between people and services by enhancing the customer experience. Chatbot is widely used in the areas of food ordering, ecommerce and transportation, etc. Practically it is not possible to find a permanent companion to make us happy all the time. Hence, this paper has planned to design a jollity chatbot to talk with the human users and make sure that it entertains and give suggestion and motivation in tough times. The jollity chatbot is implemented in Rasa, an open-source conversational AI framework and it is easy to customize. The proposed method has added 12 intents with each more than 8 text examples constituting a total of 100 input samples in nlu.md and their response in domain.yml. The flow of interactions is given in stories.md. The jollity chatbot is deployed in Telegram using ngrok and the server URL details and the access token are given in the credentials.yml. The system is experimented with various evaluation measures like accuracy of the intents, accuracy of the stories and the confusion matrix to shows that the proposed jollity chatbot system is more robust and can identify the user intents appropriately.

Tools & Technologies:

1. RASA - The primary technology used for building the chatbot. Rasa is an open-source framework for building conversational AI.
2. Generative-based model - A generative-based model is a type of machine learning model that creates new data instances by learning and generating patterns, often used in natural language processing for tasks like text generation or image synthesis

3. Telegram - Telegram is a cloud-based messaging app with a focus on speed, security, and user privacy, offering features such as end-to-end encryption, multimedia sharing, and customizable chatbots.

Results:

1. This Jollity chatbot mitigates the problems of the depression by providing an invisible friend to the user whom they can rely on and they can also chat with the bot for the whole day.
2. This is cost-effective because they don't want to afford a costly psychological session that makes them cheerful. The jollity chatbot is implemented in Rasa and deployed in Telegram.

3. The experimental results show that the system can recognize the intents and fetch the appropriate responses with an accuracy of 90%.

4. The future scope of the jollity chatbot is to fetch the real-life examples by adding the platforms like Quora so that the user can understand the feeling of another person.

2.1.7 The impact of using pre-train word embeddings in Sinhala chatbots[7]

Abstract:

Providing conversational interfaces to IT services enable more people to access them conveniently. For maximum benefit, such interfaces need to be in the native language of the users in a community. Popularly known as chatbots, one of their critical tasks is to accurately identify the user's intention from their natural language input. This paper describes an attempt at improving this accuracy in an implementation based on the open-source RASA stack, by using Sinhala word embeddings. This approach shows improvement in both intent detection as well as the confidence in the detected intents.

Tools & Technologies:

1. Pre-trained word embedding: Pre-trained word embeddings are context-rich representations of words, learned from large corpora, capturing semantic relationships. They serve as feature vectors, mapping words to high-dimensional spaces, preserving contextual meanings. Popular models like Word2Vec, GloVe, and FastText generate embeddings, enabling transfer of pre-existing knowledge to downstream NLP tasks. This minimizes the need for extensive task-specific data. These embeddings facilitate improved performance in tasks such as sentiment analysis, named entity recognition, and machine translation by providing a foundation for understanding word semantics and contextual nuances.

Results:

1. Evaluations show that the use of pre-trained word embeddings in chatbot development leads to improvements in the accuracy of the intent classification model of the chatbots.

2. Further experiments are being carried out by increasing the number of training examples in the datasets to determine if the above results remain consistent.

3. We also expect to test the performance of the named entity extraction task with and without pre-trained word embeddings.

2.1.8 Humanizing the Chatbot with Semantics based Natural Language Generation [8]

Abstract:

This paper introduces approach made for improving the efficiency of the chatbot or artificial conversational entity used in various commercial and banking sector. Humanizing is to improving the response generation ability of the chatbot. In this work, an attempt has been made to generate more natural response for a question asked to an artificial conversational entity by using various Natural Language Processing (NLP) and Natural Language Generation (NLG) techniques. Paraphrase generation plays a main role by generating semantically similar response for a query making it more natural.

Tools & Technologies:

1. Semantics-based Natural Language Generation (NLG): Semantics-based Natural Language Generation (NLG) is a field within natural language processing (NLP) that focuses on generating human-like text by understanding and incorporating the meaning, or semantics, of the underlying data. NLG systems aim to transform structured information or data into coherent and contextually appropriate natural language text.

Key components of semantics-based NLG include:

i. Semantic Representation: The input data is often represented in a structured semantic form, such as knowledge graphs or ontologies, which explicitly capture the relationships and meanings within the data.

ii. Semantic Analysis: The NLG system analyses the semantic representation of the input data to understand the underlying meaning, relationships, and context.

iii. Content Planning: The system plans the content to be generated, taking into account the identified semantics and the desired output structure. This involves determining the order of information, selecting relevant details, and organizing the content for coherent communication.

iv. Linguistic Realization: The NLG system translates the planned content into natural language text, ensuring that the generated sentences are grammatically correct, stylistically appropriate, and semantically faithful to the underlying data.

v. Context Awareness: Semantics-based NLG systems often incorporate context awareness to adapt the generated output to the specific context in which it will be presented. This includes considering user preferences, the communication medium, and any situational context.

Results:

1. Evaluations show that the use of pre-trained word embeddings in chatbot development leads to improvements in the accuracy of the intent classification model of the chatbots.

2. Further experiments are being carried out by increasing the number of training examples in the datasets to determine if the above results remain consistent.

2.2 KEY GAPS IN LITERATURE

2.2.1 Chatbot using RASA [1]

1. Dependency on External APIs: The chatbot's functionality relies on external APIs for weather and stock data. Any issues with these APIs can affect the chatbot's performance.

2. Training Data: To improve chatbot performance, extensive training data is required. The chatbot's accuracy and ability to handle various user inputs may be limited without sufficient training data

3. Complex Queries: Handling highly complex or domain-specific queries might be challenging for the chatbot, as it may require specialized knowledge beyond its training data.

4. Customization: While Rasa provides flexibility for customization, developing and fine-tuning custom actions and behaviours can be time-consuming and may require Python programming expertise.

5. Scalability: Scaling the chatbot to handle a large number of concurrent users may require additional infrastructure and optimization efforts.

6. User Experience: The chatbot's success relies on its ability to provide meaningful and accurate responses. Inaccurate or irrelevant responses can lead to a poor user experience.

7. Overall, this chatbot demonstrates the capabilities of the Rasa framework for building AI-driven conversational interfaces while highlighting some limitations related to data, external dependencies, and customization complexity.

2.2.2 Implementation of an Educational Chatbot using Rasa Framework [2]

1. Limited Data: The accuracy of the chatbot heavily depends on the training data. Limited data for training could result in lower accuracy in understanding user intents and entities.

2. Training Time: Training deep learning models, such as those used in Rasa, can be time-consuming and may require substantial computational resources.

3. User Adoption: The paper briefly mentions that many people may not come forward to use certain applications, such as those designed for visually impaired individuals. User adoption and training could be challenges.

4. Language Support: The chatbot appears to be designed for English language interaction. Expanding language support may be necessary for broader accessibility.

5. Chatbot Maturity Level: The paper does not specify the complexity level of the chatbot (e.g., whether it's at Level 1, Level 2, etc., as mentioned in the introduction), which could affect its effectiveness.

6. Discoverability: Discoverability of the chatbot on social media platforms is noted as a challenge, as it requires text processing and training for accurate results.

7. Accuracy: The paper mentions the need for improving the accuracy of intent and entity recognition. This implies that the current accuracy may not be optimal.

2.2.3 RASA Chatbot Using AI [3]

The paper does not extensively discuss limitations, but it implies several potential challenges:

1. Training Data: Like many AI systems, the effectiveness of the chatbot depends on the quality and quantity of training data. Inadequate data can lead to lower accuracy in understanding user queries.
2. Complexity: While the paper highlights the capabilities of the Rasa framework, building complex chatbots can require a team of experts and substantial development effort.
3. Resource Scarcity: The paper mentions the use of open-source software to overcome resource scarcity, indicating that limited resources may be a constraint for chatbot development.
4. Customization: Customizing chatbots may require a deep understanding of the framework's internals, which could be a challenge for developers who are not familiar with natural language processing.
5. Voice and Face Recognition: While mentioned as a future scope, integrating voice and face recognition can be technically challenging and resource-intensive.
6. User Adoption: The paper doesn't discuss user adoption and how readily users accept and use the chatbot, which can be a critical factor in its success.
7. Maintenance: While it mentions low maintenance as a benefit, maintaining the chatbot's performance and adding new features over time can be an ongoing task.
8. Privacy and Security: The paper doesn't explicitly address issues related to user data privacy and security, which are essential considerations in chatbot development.

2.2.4 College Enquiry Chatbot using RASA [4]

While the paper does not explicitly mention limitations, some potential limitations of the chatbot system could include:

1. Limited Knowledge Base: The chatbot's effectiveness heavily relies on the accuracy and completeness of the information stored in its database. If the knowledge base is not regularly updated, it may not provide accurate responses.
2. Dependency on Text Data: Chatbots like this one primarily rely on text input from users. Handling complex or nuanced queries, especially those requiring visual or contextual information, may be challenging.

3. Language Support: The chatbot may be limited to specific languages, and it may struggle with understanding queries in languages other than the one it's designed for.

4. Scalability: The paper doesn't discuss how easily the chatbot can scale to accommodate a larger user base. Scalability can be a significant challenge in real-world applications.

5. User Experience: The user experience can be impacted if the chatbot fails to understand or misinterprets user queries, leading to frustration among users.

6. Hardware Requirement: The hardware requirements mentioned in the paper (i3 processor-based computer and 2GB RAM) may limit the deployment of the chatbot on low-end devices.

7. Maintenance and Updates: The chatbot would require regular maintenance and
   updates to keep up with changes in college information and user needs.

2.2.5 Predicting Frequently Asked Questions (FAQs) on the COVID-19 Chatbot using the DIET Classifier [5]

1. Data Quality and Diversity: Gaps in the quality and diversity of the training data can impact the performance of the DIET Classifier. If the dataset used for training is limited in scope or not representative of various user queries, the model might struggle to generalize to a broader range of questions.

2. Dynamic Nature of FAQs: FAQs related to COVID-19 are subject to change as new information becomes available. The research might not have addressed the dynamic nature of frequently asked questions during the pandemic. An effective chatbot should be capable of adapting to evolving information and updating its responses accordingly.

3. User Intent Understanding: Predicting FAQs involves understanding user intents accurately. If the DIET Classifier fails to capture the nuanced meanings of user queries or misclassifies intents, it could lead to suboptimal chatbot performance.

4. Handling Ambiguity and Variability: User queries can be ambiguous, and the same question might be phrased in various ways. The DIET Classifier or the chatbot system should be robust enough to handle such variations and provide accurate responses.
5. Evaluation Metrics: The research might not have thoroughly explored or discussed the choice of evaluation metrics. Precision, recall, F1-score, or other relevant metrics should be considered and reported to assess the model's performance comprehensively.

2.2.6 Jollity Chatbot- A contextual AI Assistant[6]

1. Context Handling: The paper might not thoroughly address how the Jollity Chatbot handles contextual information in user interactions. An effective chatbot should be able to maintain context across multiple turns in a conversation, understanding and responding appropriately to follow-up queries.

2. User Intent Recognition: If the paper does not discuss the methods used for user intent recognition, there might be a gap in understanding and accurately categorizing user queries. An effective chatbot needs to recognize user intents to provide relevant and meaningful responses.

3. Real-world Application and Testing: A gap may exist if the research lacks details on real-world testing and application scenarios. Understanding how the Jollity Chatbot performs in practical, diverse environments is crucial for evaluating its effectiveness and usability.

4. Multimodal Interaction: If the research focuses only on text-based interactions and neglects multimodal aspects (such as handling images, voice inputs, or other forms of data), it might not capture the full potential of a contextual AI assistant.

5. User Experience Evaluation: The paper may not delve deeply into the user experience aspect, including user satisfaction, ease of use, and user preferences. Evaluating the chatbot's effectiveness from the user's perspective is essential for its success.

2.2.7 The impact of using pre-trained word embeddings in Sinhala Chatbots[7]

1. Domain Specificity: The research may not address the impact of pre-trained word embeddings in domain-specific contexts. Chatbots often serve different domains, and the effectiveness of embeddings might vary across these domains.

2. Handling Out-of-Vocabulary Words: If the paper does not discuss how the Sinhala chatbot handles out-of-vocabulary words, there might be a gap in understanding its robustness to novel terms or evolving language use.
3. User Interaction and Experience: A gap could exist if the study does not thoroughly explore the impact of pre-trained word embeddings on user interaction and experience in the context of Sinhala-speaking users.

4. Resource Requirements: The paper may not address the computational and resource requirements associated with using pre-trained word embeddings. Understanding the trade-offs in terms of computational efficiency is crucial, especially for practical deployment.

2.2.8 Humanizing the Chatbot with Semantics based Natural Language Generation [8]

1. Definition and Measurement of Humanization: If the research lacks a clear definition of what is meant by "humanizing" a chatbot and how this humanization is measured, there might be a gap in understanding the key aspects being addressed.

2. User-Centric Evaluation: A gap might exist if the study does not thoroughly explore user-centric evaluations of the humanization efforts. Understanding how users perceive and interact with the chatbot is crucial for assessing the success of humanization strategies.

3. Diversity in Language and Culture: If the paper does not discuss how humanization strategies consider linguistic and cultural diversity, there might be a gap in understanding the generalizability of these strategies across different user groups.

4. Real-Time Adaptability: A gap could exist if the research does not explore the chatbot's ability to adapt its humanization strategies in real-time based on user feedback or changes in the conversation context.

5. Handling Sarcasm and Humour: If the study does not address how the chatbot handles sarcasm, humour, or other nuanced aspects of human communication, there might be a gap in the chatbot's ability to truly humanize interactions.

6. Long-Term User Engagement: The paper may not discuss the impact of semantics-based natural language generation on long-term user engagement. Understanding how humanization contributes to sustained user interest is crucial for the success of a chatbot.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 REQUIREMENT AND ANALYSIS

It is very important to carefully check the requirements and do a full study of a Rasa chatbot project. When gathering and studying requirements, it's important to keep the following important things in mind:

1. Objectives of the business:
    - Articulate the specific objectives that the Rasa chatbot is intended to achieve for the firm.
    - Identify the specific scenarios and circumstances in which the chatbot will be employed.
2. User requirements:
    - Identify the user personas and the target audience.
    - Identify the primary goals and anticipated outcomes of people engaging with the chatbot
    - Consider the user's preferences for various communication channels, such as web platforms and messaging applications.

3. Functional Requirements:
    - Comprehending Natural Language (NLU):
        - Provide a description of the relevant entities and supported intents inside the domain.
        - Specify the required degree of understanding of natural language, such as the ability to handle variations in user input.
    - Dialogue Management:
        - Provide an explanation of the decision trees and conversation flows that pertain to various user engagements.
        - Define the protocol for the chatbot to handle multi-turn discussions.

- Incorporating External Systems:
    - Identify the external systems, such as databases or third-party services, that the chatbot must integrate with.
    - Specify the kind of data retrieval or transactions that the chatbot will manage.
- Specific Actions:
    - Enumerate the specific custom activities, such as executing API calls and modifying databases, that the chatbot should be capable of performing.
    - User Authorization and Authentication:
    - Specify the requirements for user authorization and authentication, if applicable.
- Support for multiple languages:
    - Assess the necessity of providing multilingual support for the chatbot.

4. Non-Functional Requirement :
    - Scalability:
        - Provide the required level of scalability and the expected user workload
    - Security:
        - Implement robust security protocols to safeguard user data and prevent unauthorized intrusion.
        - Ensure compliance with relevant data protection legislation.
    - Accomplishment:
        - Set benchmarks for system latency and response times.
    - Reliability:
        - Specify your requirements about reliability and the amount of time the system should be operational.
    - Observation and analysis:
        - Enumerate the metrics that will be monitored, such as the number of successful interactions and the level of user involvement.
        - Select the platforms or technologies to be utilized for reporting and analytics.

5. Integration Requirements:
    - Identify the existing platforms, applications, or systems to enable seamless integration of the chatbot with them.
    - Determine the formats and protocols utilized for data interchange.

6. Management and training of the model:
    - The process of acquiring and classifying training data for neural language models involves several steps.
    - Determine the protocol and regularity for updating models with new data.

7. Compliance with Regulations and Protocols:
    - Ensure that the chatbot complies with all relevant laws and regulations in the given locations or sectors.

8. Creating User Experience (UX):
    - Specify the design principles and desired user interaction.
    - Consider the congruity of the brand, the style of communication, and the visual elements.

9. Maintenance and Support:
    - Outline the protocol for routine updates, assistance, and upkeep.
    - Specify the pathways for reporting problems and obtaining user input.

10. Evaluation of user experience and creation of preliminary models:
    - In order to assess the functionality and user experience of the chatbot, it would be advisable to consider creating a prototype.
    - Schedule user testing to solicit feedback and continuously enhance your product.

3.2 PROJECT DESIGN AND ARCHITECTURE

The process of designing the architecture for a Rasa chatbot entails establishing the framework of the different components and specifying how they interact with each other. Fig 1 presents the architectural requirements for our Rasa chatbot. User Interface (UI): The graphical or textual interface via which users engage with the chatbot, accessible via web interfaces or messaging services.

Following are the components in the Rasa architecture. A user interface can be developed on various platforms and we have created a UI using basic HTML but the same can be integrated on the web or a messaging platform such as Slack or Facebook Messenger.

1 Rasa NLU (Natural Language Understanding) is a system that analyzes and comprehends user input, extracting the intended meaning and identifying specific elements or concepts spoken.

   Constituents:

   - Intent Classifier: Determines the user's intention.
   - Entity Extractor: Retrieves pertinent entities from user input.
   - NLU Training Data: Comprises annotated instances used to train the NLU model.

2 Rasa Core: Function: Oversees the progression of the conversation and determines the subsequent course of action based on the user's input.

   Constituents:

   - Dialogue Management: Determines the sequence of conversation.
   - Custom Actions: Specifies the actions that the bot is capable of executing.
   - Stories: Dataset used to train dialogue sequences.
   - Policies: Establish the guidelines for forecasting the subsequent course of action.

3 External Systems: Incorporation of other systems, including databases, APIs, and other services.

   Components: Banking

   - APIs: Establishes connections with fundamental banking systems.
   - External Services: Establishes connections with external service providers.

4 Action Server: Manages and executes custom actions specified in Rasa Core.
Components: Custom Action Handlers: Perform activities such as invoking APIs or modifying databases.

5 Data Storage: Serves as a repository for durable data, such as user profiles or chat history.
Components: Database: Stores data pertaining to users and the history of their conversations.

6 Monitoring and Analytics: Description: This component observes and analyzes user interactions, system performance, and analytics to identify areas for improvement.
Constituents: The Analytics Module is responsible for gathering and examining data related to user interactions.

7 Environment for Deployment: The location and availability of the chatbot for users.

- Web Server: Provides the web interface.

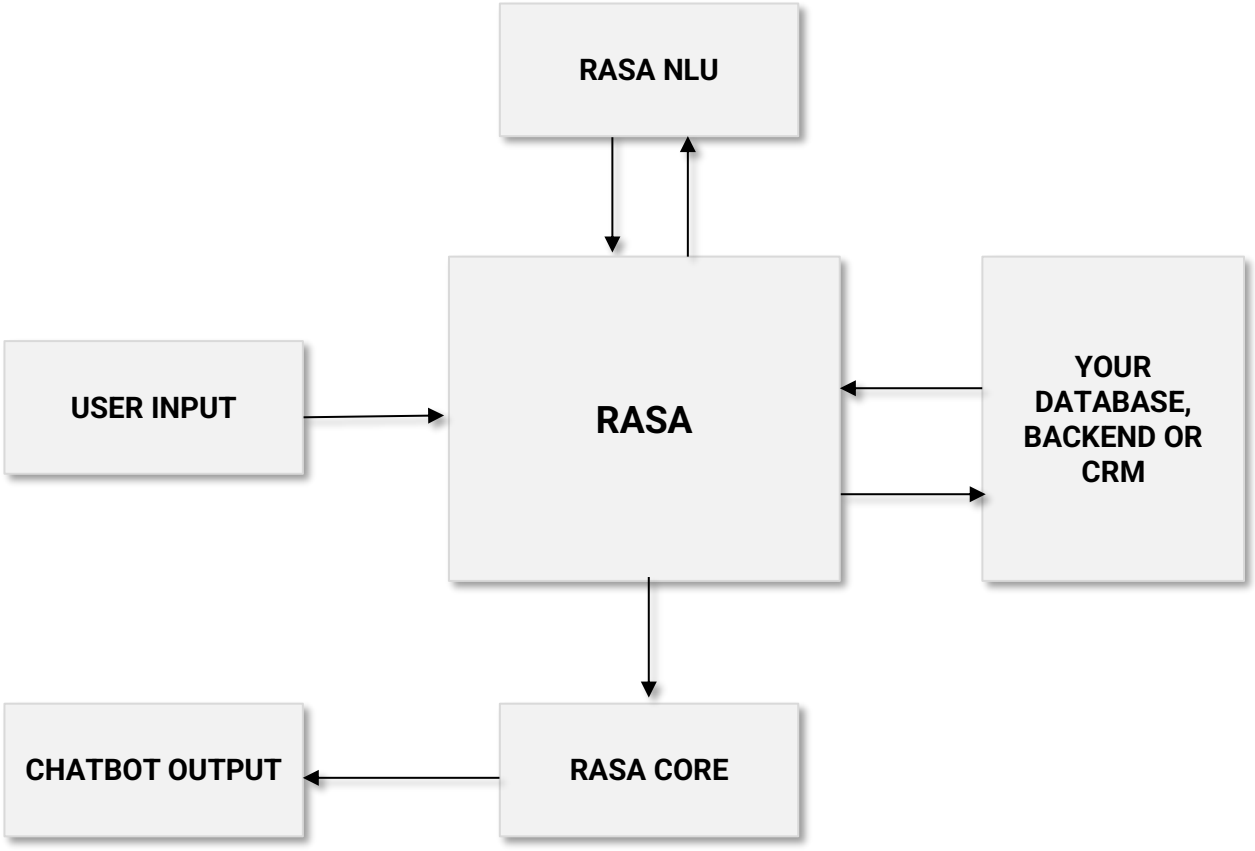


Fig 1: Data flow diagram for the working of a RASA Chatbot

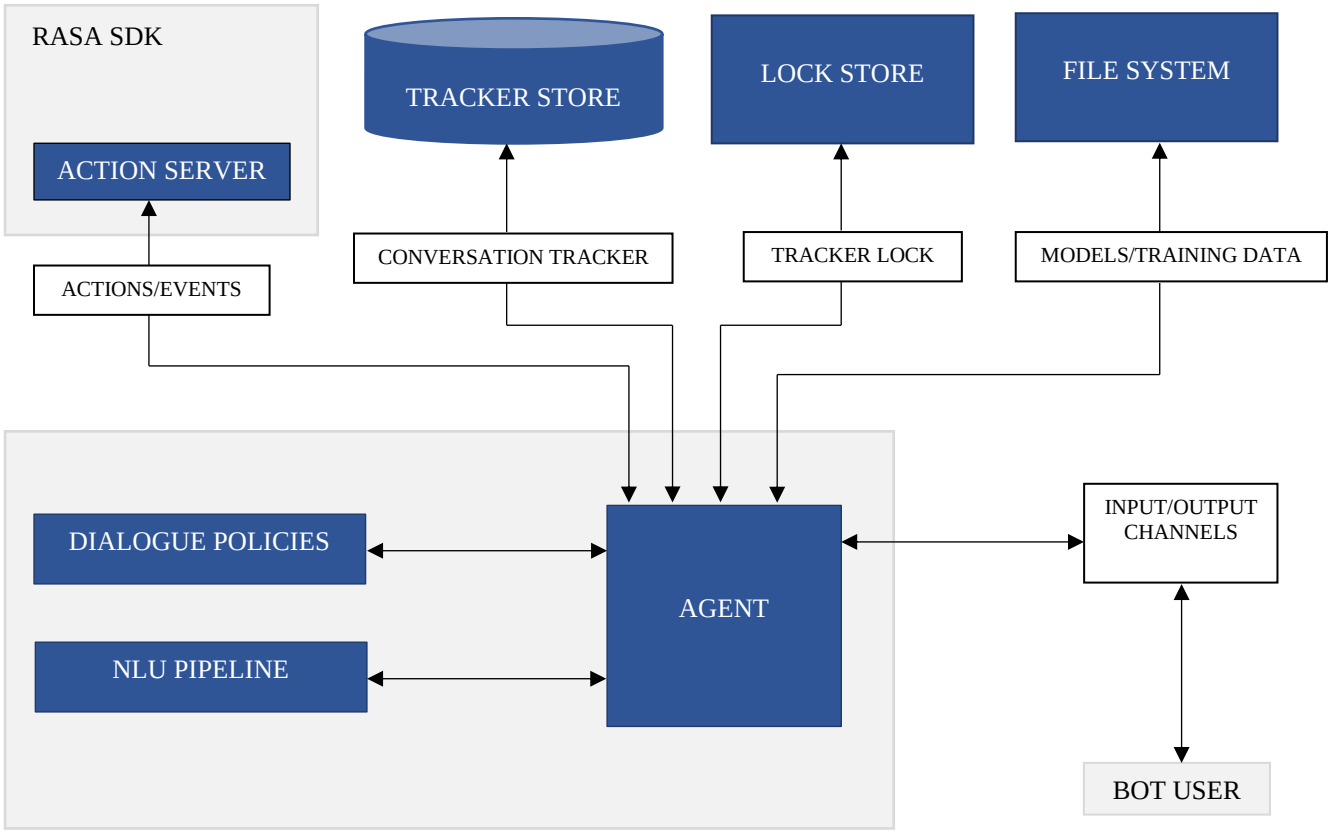


Fig 2: The diagram above shows Rasa architecture in detail.

Fig 2 Represents rasa architecture where we can see NLU and dialogue management as the major components. NLU includes intent categorization, entity extraction, and response retrieval. The NLU Pipeline processes user utterances through an NLU model using a learned pipeline. The dialogue management component analyzes the circumstances to decide what to say next. The figure shows Dialogue Policies.

3.3 DATA PREPARATION

Data preparation is an essential and pivotal stage in constructing a Rasa chatbot, as it entails the creation of top-notch training data for both the Natural Language Understanding (NLU) and Core components. Below are essential points to consider when preparing data for a Rasa chatbot:

- Understand Your Domain: Develop an extensive comprehension of the specific field in which the chatbot will function. Comprehend the many categories of inquiries, purposes, and elements that are pertinent to the users.
- Define Intents and Entities: Precisely specify the intents, which represent the goals of the user, and the entities, which are the pertinent information that the chatbot should be capable of identifying. In the context of a banking chatbot, examples of intents could be "retrieve account balance" or "initiate funds transfer, " whereas entities could refer to "account category" or "transaction amount."
- Collect Diverse Examples: Ensure that the training data encompasses a diverse array of methods via which users may articulate the identical intention.
- Handle Synonyms and Variations: Incorporate many methods that users may employ to convey the same intention or offer comparable things.
- Use Utterances from Real Users: to enhance the authenticity of the training data. This enhances the model's ability to generalize more effectively to user inputs that it has not encountered before.
- Endpoint Configuration: Define the endpoints for your locally hosted API. Decide on the route and port where your API will be accessible locally.
- Security Measures: Implement security measures such as API keys or authentication to control access to your API and prevent unauthorized usage.
- Error Handling: Develop robust error handling mechanisms to gracefully manage and respond to errors that may occur during API requests.
- Input Validation: Validate input data to ensure that it meets the expected format and constraints, preventing potential issues and vulnerabilities.

- CORS Policy: Set up Cross-Origin Resource Sharing (CORS) policies to specify which origins are allowed to access your API, enhancing security and controlling access from web browsers.
- Documentation: Create comprehensive documentation for your API, including details on endpoints, request/response formats, and usage examples, to facilitate its integration and usage by other developers. Define Stories for Dialogue Management: Generate dialogue stories for the purpose of training the Rasa Core component. Stories depict simulated dialogues between the user and the chatbot, outlining the chronological order of user inputs and anticipated bot responses.

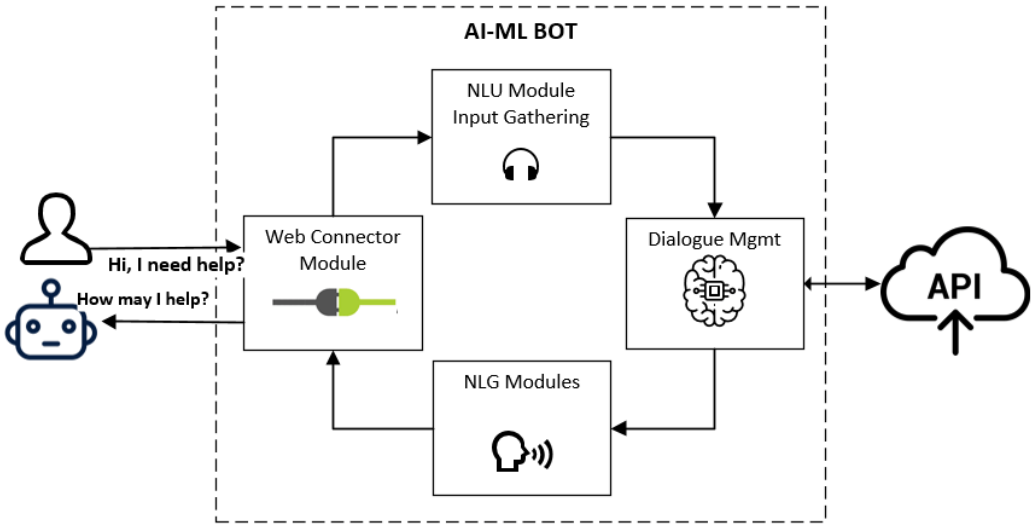


Fig 3: The diagram describes how RASA bot uses API call to fetch information based on user input

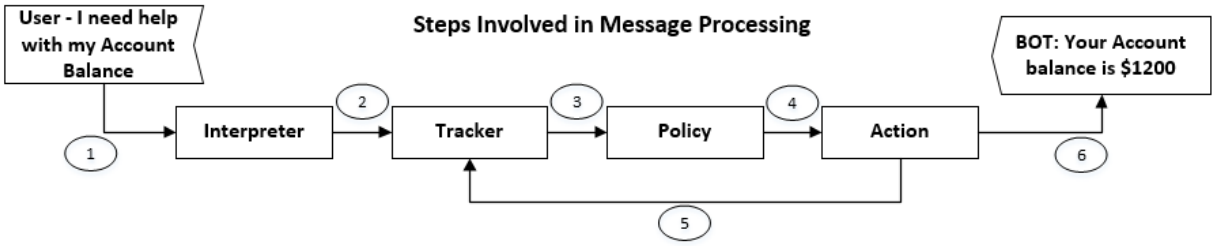


Fig 4: Steps involved in NLU when any new message processing.

3.4 IMPLEMENTATION

In a Rasa chatbot project, various algorithms, tools, and techniques are employed for natural language understanding, dialogue management, and overall system development. Here's an overview of key elements:

- Natural Language Understanding (NLU):
  - Algorithm:

    Rasa relies on various machine learning algorithms for NLU, including SVM (Support Vector Machines) and TensorFlow for intent classification and entity recognition.
  - Tools and Techniques:

    Rasa NLU: A library for natural language processing and understanding that is part of the Rasa framework.

    Spacy: An open-source library for advanced natural language processing.
- Dialogue Management:
  - Algorithm:

    Rasa Core uses a form of deep reinforcement learning, specifically a policy-based approach, for dialogue management.
  - Tools and Techniques:

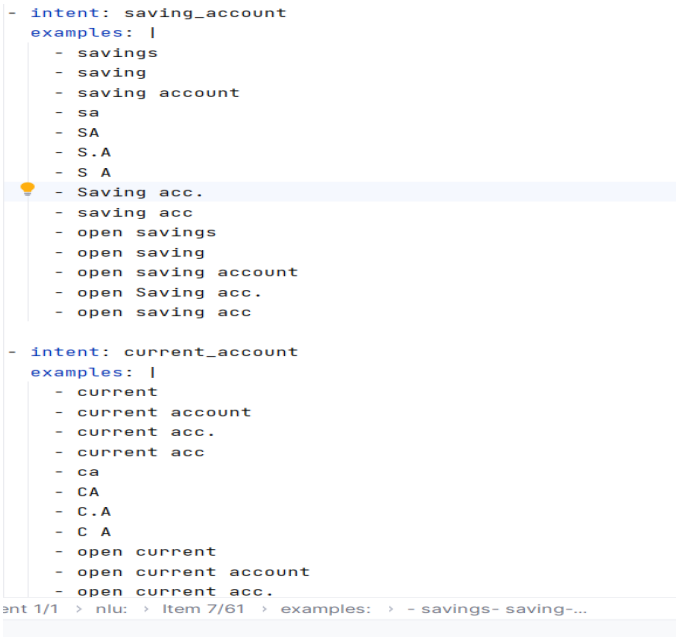
    Rasa Core: The dialogue management component in the Rasa framework.

    Policies: Configurable components that define the behavior of the dialogue management model.
- Custom Actions:

  Rasa Core's custom actions allow developers to define custom behavior for the chatbot, such as calling APIs or interacting with external systems.
- Model Training and Evaluation:

  Rasa Train: Command-line tool for training Rasa NLU and Core models.

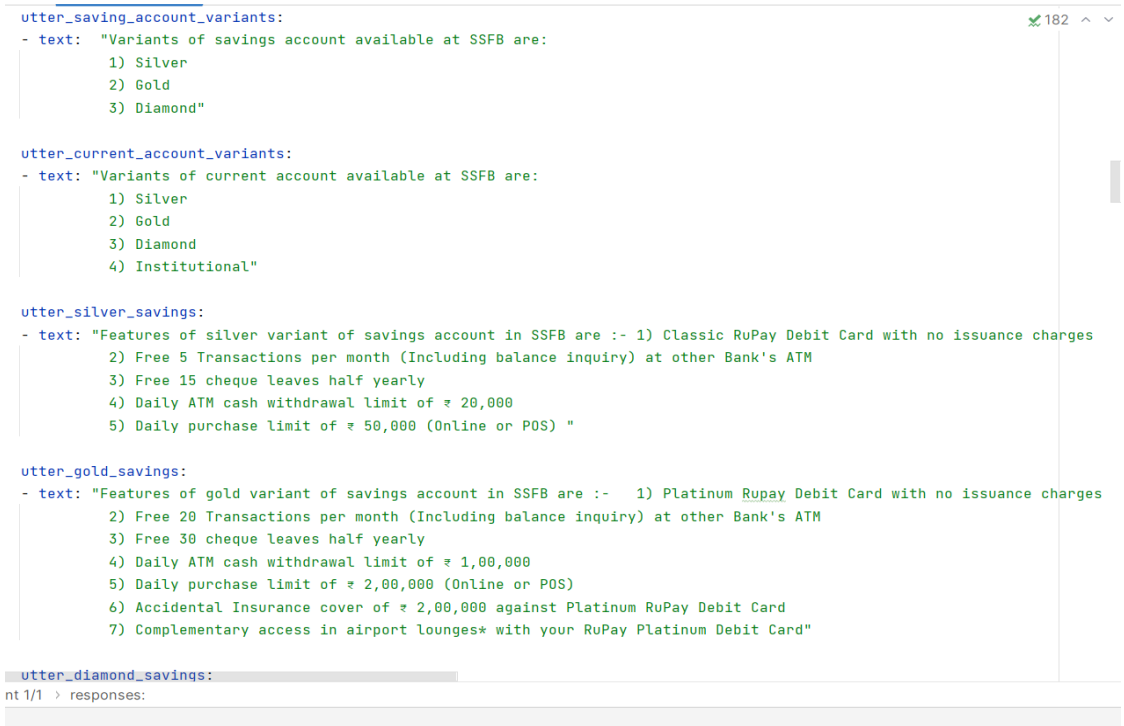
  Cross-Validation: Splitting data into training and test sets to evaluate model performance.
- Integration with External Systems: Web pages , applications etc.



```
- intent: saving_account
  examples: |
    - savings
    - saving
    - saving account
    - sa
    - SA
    - S.A
    - S A
    - Saving acc.
    - saving acc
    - open savings
    - open saving
    - open saving account
    - open Saving acc.
    - open saving acc

- intent: current_account
  examples: |
    - current
    - current account
    - current acc.
    - current acc
    - ca
    - CA
    - C.A
    - C A
    - open current
    - open current account
    - open current acc.
ent 1/1 › nlu: › Item 7/61 › examples: › - savings- saving-...
```

Fig 5: Implementation of NLU file by creating various intents.

Fig 5 is a code snippet of the NLU file in which is the the intents for saving account and current account is mentioned



```
utter_saving_account_variants:
- text:  "Variants of savings account available at SSFB are:
         1) Silver
         2) Gold
         3) Diamond"

utter_current_account_variants:
- text: "Variants of current account available at SSFB are:
         1) Silver
         2) Gold
         3) Diamond
         4) Institutional"

utter_silver_savings:
- text: "Features of silver variant of savings account in SSFB are :- 1) Classic RuPay Debit Card with no issuance charges
         2) Free 5 Transactions per month (Including balance inquiry) at other Bank's ATM
         3) Free 15 cheque leaves half yearly
         4) Daily ATM cash withdrawal limit of ₹ 20,000
         5) Daily purchase limit of ₹ 50,000 (Online or POS) "

utter_gold_savings:
- text: "Features of gold variant of savings account in SSFB are :-   1) Platinum Rupay Debit Card with no issuance charges
         2) Free 20 Transactions per month (Including balance inquiry) at other Bank's ATM
         3) Free 30 cheque leaves half yearly
         4) Daily ATM cash withdrawal limit of ₹ 1,00,000
         5) Daily purchase limit of ₹ 2,00,000 (Online or POS)
         6) Accidental Insurance cover of ₹ 2,00,000 against Platinum RuPay Debit Card
         7) Complementary access in airport lounges* with your RuPay Platinum Debit Card"

utter_diamond_savings:
nt 1/1 › responses:
```

Fig 6: Implementation of domain file by putting text responses as utterances

Fig 6 is a code snippet of the utterances that have been coded in the domain file of the RASA project which will be displayed as a response in our chatbot.

```yaml
- story: neft path 1
  steps:
  - intent: neft_transfer
  - action: utter_neft_transfer
  - intent: affirm
  - action: utter_neft_process
  - intent: affirm
  - action: utter_neft_eligibility
  - intent: affirm
  - action: utter_neft_limits

- story: neft path 2
  steps:
  - intent: neft_process
  - action: utter_neft_process
  - intent: affirm
  - action: utter_neft_eligibility
  - intent: affirm
  - action: utter_neft_limits

- story: neft path 3
  steps:
  - intent: neft_eligibility
  - action: utter_neft_eligibility
  - intent: affirm
  - action: utter_neft_limits

- story: neft path 4
  steps:
  - intent: neft_limits
```

ent 1/1 › stories: › Item 37/76 › steps:

Fig 7: Creating story pathways in the stories file.

Fig 7 is a code snippet where the story pathways have been created in the yml file for stories.

```yaml
socketio:
 user_message_evt: user_uttered
 bot_message_evt: bot_uttered
 session_persistence: true
#mattermost:
#  url: "https://<mattermost instance>/api/v4"
#  token: "<bot token>"
#  webhook_url: "<callback URL>"

# This entry is needed if you are using Rasa Enterprise. The entry represents credentials
# for the Rasa Enterprise "channel", i.e. Talk to your bot and Share with guest testers.
rasa:
  url: "http://localhost:5002/api"
```

Fig 8 Snippet of the credentials

Fig 8 is code snippet of the file where configuration of the communication channel has being done through Socket.IO and specifying the URL for Rasa.

3.5 KEY CHALLENGES

The creation of a Rasa chatbot involves a range of challenges, including activities such as natural language understanding and system integration. Addressing these challenges is crucial for building a robust and effective conversational agent.

1. Training Data Quality:
    - Challenge: Insufficient or low-quality training data leading to poor model performance.
    - Solution: Invested more time in creating diverse and representative training data. Trying to cover ambiguity and achieve a good data set.
2. Intent Ambiguity:
    - Challenge: Users may express intents in various ways, leading to ambiguity.
    - Solution: Augmented training data with a variety of user expressions .Implemented context-aware dialogue management to disambiguate user intent over multiple turns.
3. Entity Recognition:
    - Challenge: Accurately extracting entities from user input can be challenging, especially with complex entities.
    - Solution: Used tools like spaCy for custom entity recognition. And regularly reviewed and refined entity annotation in training data.
4. Handling Out-of-Domain Queries:
    - Challenge: Users might pose queries unrelated to the chatbot's domain.
    - Solution: Implemented a robust fallback mechanism to gracefully handle out-of-domain queries.
5. Model Training and Evaluation:
    - Challenge: Balancing model complexity and overfitting during training.
    - Solution: Regularly evaluated model using test datasets. Fine-tuned hyperparameters and employed techniques like cross-validation to prevent overfitting.

CHAPTER 4

TESTING

4.1 TESTING STRATEGIES

Testing is a critical aspect of developing a Rasa chatbot to ensure its functionality, accuracy, and reliability. Here are testing strategies for a Rasa chatbot:

1. Unit Testing:
    - Objective:

      Verify the correctness of individual components such as custom actions, utility functions, and NLU models.
    - Techniques:

      Write tests for custom actions to ensure they produce the expected responses. Test NLU models with labelled examples to confirm accurate intent classification and entity extraction.
2. Integration Testing:
    - Objective:

      Validate the interactions and collaboration between different components, including the Rasa Core and NLU components.
    - Techniques:

      Simulate end-to-end conversations to ensure proper integration between NLU and Core components. Verify that custom actions interact correctly with external systems.
3. End-to-End Testing:
    - Objective:

      Validate the entire chatbot's functionality, covering both NLU and Core components.
    - Techniques: Define test scenarios that mimic real user interactions.

      Use tools like Rasa Test to automate end-to-end tests and evaluate bot responses.

4. Regression Testing:
    - Objective:

      Ensure that new changes or updates do not introduce regressions in existing functionality.
    - Techniques:

      Re-run previously successful tests after making updates to the chatbot.

      Maintain a test suite that covers critical user journeys.
5. User Simulation Testing:
    - Objective:

      Simulate various user personas and scenarios to assess the chatbot's ability to handle diverse inputs.
    - Techniques:

      Create test cases that mimic different user behaviours and intents.

      Assess how well the chatbot adapts to variations in user input.
6. Load Testing:
    - Objective:

      Evaluate the chatbot's performance and responsiveness under varying levels of user load.
    - Techniques:

      Use tools like Locust or Apache JMeter to simulate concurrent users and measure response times.

      Identify and address bottlenecks for scalability.
7. Security Testing:
    - Objective:

      Assess the chatbot's security measures, including the handling of sensitive information.
    - Techniques:

      Review and test encryption mechanisms for data in transit.

      Assess the chatbot's ability to handle malicious inputs and potential vulnerabilities.

8. API Integration Testing:
    - Objective:

      Verify the seamless integration and functionality of external APIs utilized by the chatbot, such as database connections, third-party services, or custom APIs.
    - Techniques:

      Perform tests to ensure correct data retrieval, storage, and manipulation through API calls.

      Validate error handling mechanisms for cases such as network failures or invalid responses from external APIs.

      Use mocking or stubbing techniques to simulate API responses and test different scenarios, including edge cases and error conditions.

      Monitor and analyze API response times and performance metrics to ensure optimal integration and responsiveness.

      Conduct compatibility testing to ensure compatibility with different API versions, protocols, and dependencies.

      Implement API contract testing to verify adherence to specified API contracts and prevent regressions in integration functionality.

4.2 TEST CASES AND OUTCOMES

The testing process aimed to evaluate the functionality, accuracy, and user experience of the banking Rasa bot. Different test cases covered aspects such as intent and entity recognition, multi-turn conversations, error handling, and security. Here are the key findings:

1. Intent and Entity Recognition

   Outcome: The bot accurately recognized user intents and extracted entities in most scenarios.

   Interpretation: The intent and entity recognition components of the bot are robust, successfully handling various user queries related to account management, transactions, and inquiries about account types.

2. Multi-turn Conversations

   Outcome: The bot demonstrated effective handling of multi-turn conversations, providing coherent responses across consecutive user inputs.

   Interpretation: The dialogue management system successfully retained context and engaged users in meaningful conversations, enhancing the overall user experience.

3. Fallback Mechanism

   Outcome: The fallback mechanism appropriately handled out-of-scope queries, redirecting users to relevant banking tasks.

   Interpretation: The fallback mechanism effectively guides users back to the main banking functionalities, maintaining a focused user experience.

4. Error Handling

   Outcome: The bot accurately identified and communicated errors, guiding users to correct inputs.

   Interpretation: The error handling mechanisms are robust, contributing to a user-friendly experience by providing clear instructions and preventing incorrect transactions.

CHAPTER 5

RESULTS AND EVALUATION

5.1 RESULTS

Interpreting the results of testing for a banking Rasa bot involves analysing the performance across various test cases. A selection of the results obtained in accordance with the test cases mentioned earlier can be found in the following snippets

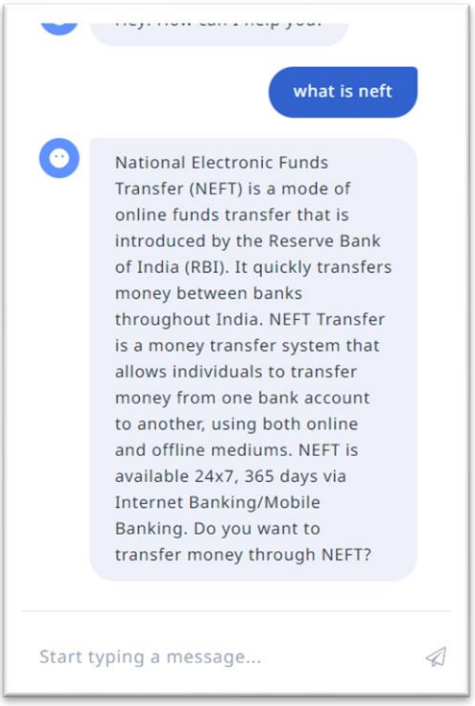


Fig 9 Intent and Entity Recognition

Fig 9 Representation of correct intent (NEFT) picked and mapped to provide correct information regarding NEFT

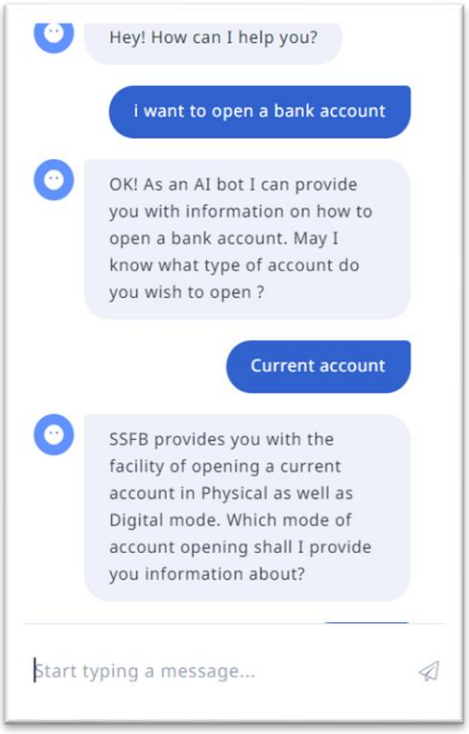


Fig 10 Results for contextual awareness

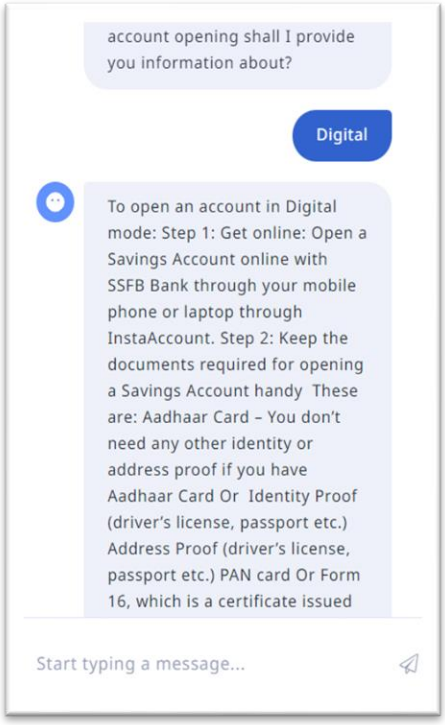


Fig 11 Result for contextual awareness

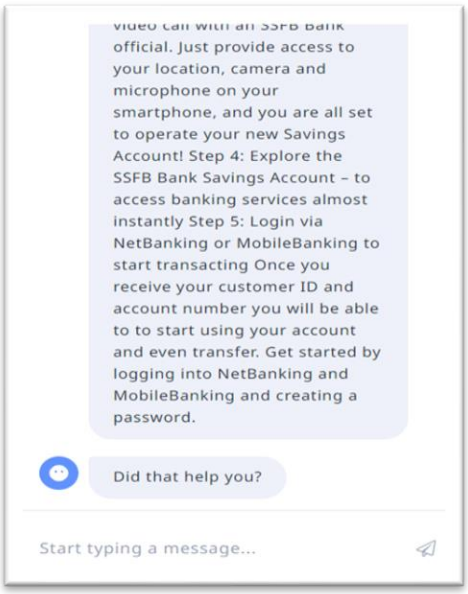


Fig 12 Result for contextual awareness

Fig 10,11,12 are the representation of how RASA bot can showcase contextual awareness and respond accordingly, providing us with a multiturn chatbot

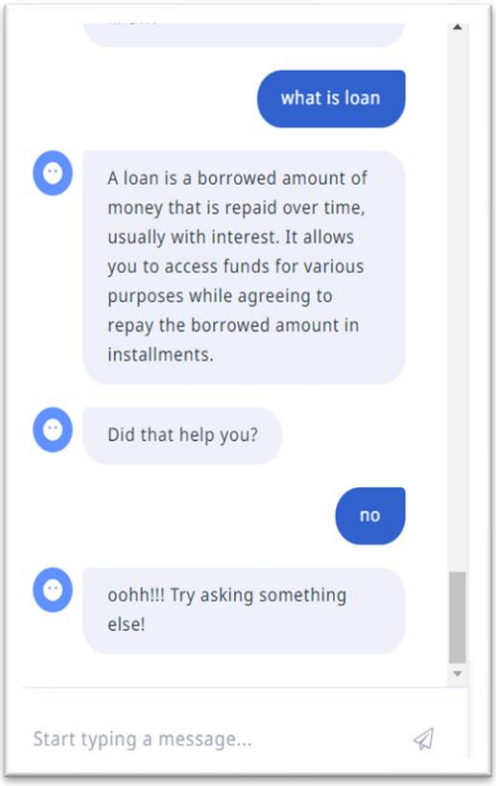


Fig 13 Proper representation of the fallback mechanism on encountering the response “no”

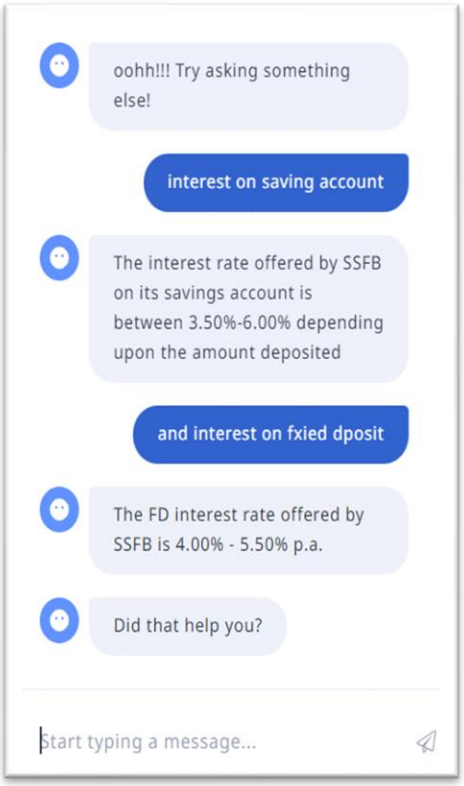
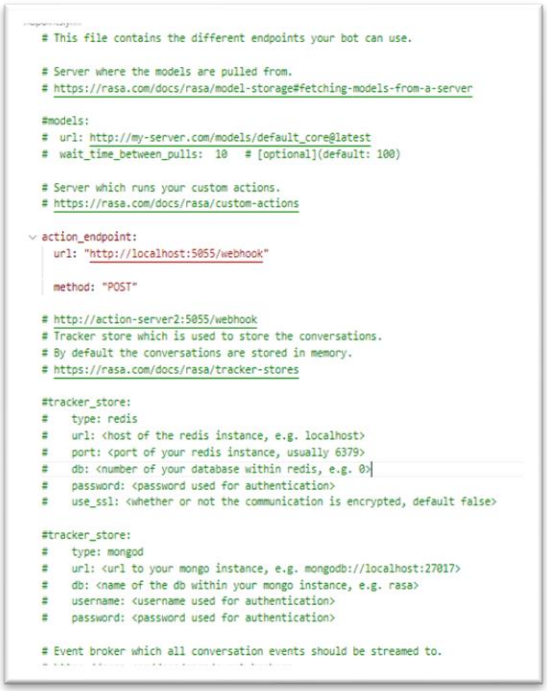


Fig 14 Proper representation of handling errors, in the above snippet the g grammatical error “fxied dposit” has been understood by the bot

.

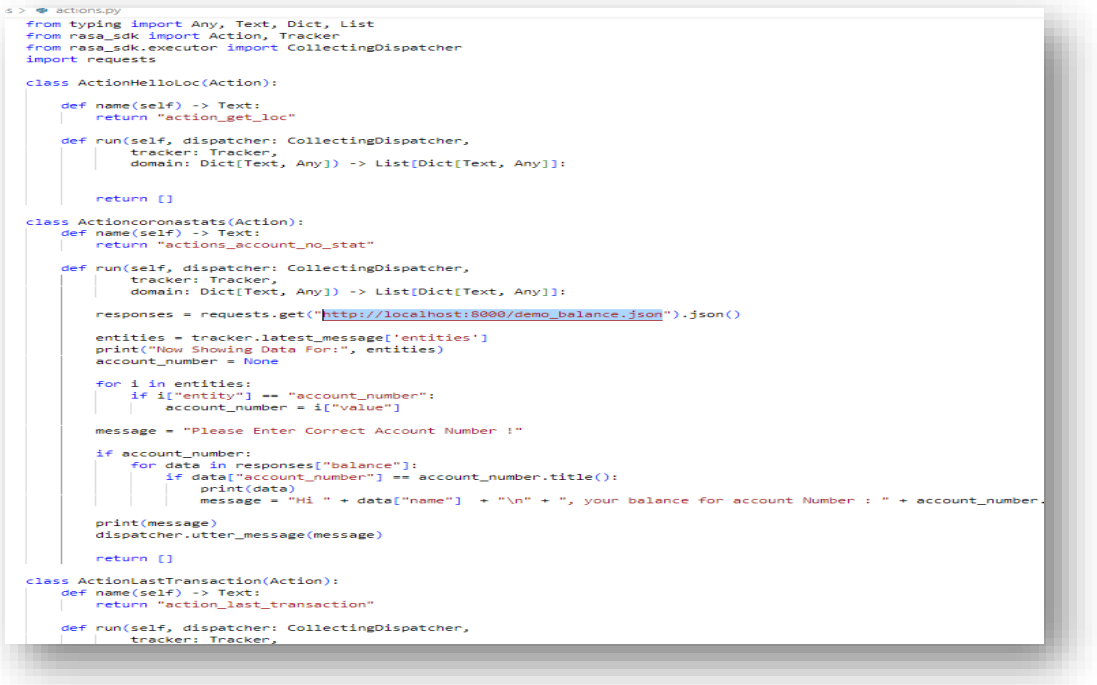
```
ersion: "3.1"

rule: activate balance loop
steps:
- intent: know_bank_balance
- action: balance_form
- active_loop: balance_form

rule: submit balance
condition:
- active_loop: balance_form
steps:
- action: balance_form
- active_loop: null
- slot_was_set:
  - requested_slot: null
- action: actions_account_no_stat


rule: activate last transaction loop
steps:
- intent: last_transactions
- action: last_transaction_form
- active_loop: last_transaction_form

rule: submit last transaction
condition:
- active_loop: last_transaction_form
steps:
- action: last_transaction_form
- active_loop: null
- slot_was_set:
  - requested_slot: null
- action: action_last_transaction


rule: activate fd summary loop
steps:
- intent: fd summary
```

Fig 15 Set of rules defined to trigger some custom actions.



```
# This file contains the different endpoints your bot can use.

# Server where the models are pulled from.
# https://rasa.com/docs/rasa/model-storage#fetching-models-from-a-server

#models:
#  url: http://my-server.com/models/default_core@latest
#  wait_time_between_pulls:  10   # [optional](default: 100)

# Server which runs your custom actions.
# https://rasa.com/docs/rasa/custom-actions

action_endpoint:
  url: "http://localhost:5055/webhook"

  method: "POST"

# http://action-server2:5055/webhook
# Tracker store which is used to store the conversations.
# By default the conversations are stored in memory.
# https://rasa.com/docs/rasa/tracker-stores

#tracker_store:
#    type: redis
#    url: <host of the redis instance, e.g. localhost>
#    port: <port of your redis instance, usually 6379>
#    db: <number of your database within redis, e.g. 0>
#    password: <password used for authentication>
#    use_ssl: <whether or not the communication is encrypted, default false>

#tracker_store:
#    type: mongod
#    url: <url to your mongo instance, e.g. mongodb://localhost:27017>
#    db: <name of the db within your mongo instance, e.g. rasa>
#    username: <username used for authentication>
#    password: <password used for authentication>

# Event broker which all conversation events should be streamed to.
```

Fig 16 Endpoints file to set the actions file endpoints



```
s > actions.py
from typing import Any, Text, Dict, List
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
import requests

class ActionHelloLoc(Action):

    def name(self) -> Text:
        return "action_get_loc"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:


        return []

class Actioncoronastats(Action):
    def name(self) -> Text:
        return "actions_account_no_stat"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        responses = requests.get("http://localhost:8000/demo_balance.json").json()

        entities = tracker.latest_message['entities']
        print("Now Showing Data For:", entities)
        account_number = None

        for i in entities:
            if i["entity"] == "account_number":
                account_number = i["value"]

        message = "Please Enter Correct Account Number !"

        if account_number:
            for data in responses["balance"]:
                if data["account_number"] == account_number.title():
                    print(data)
                    message = "Hi " + data["name"]  + "\n" + ", your balance for account Number : " + account_number.

        print(message)
        dispatcher.utter_message(message)

        return []

class ActionLastTransaction(Action):
    def name(self) -> Text:
        return "action_last_transaction"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
```

Fig 17 Actions file for custom actions that will fetch some information from external sources.

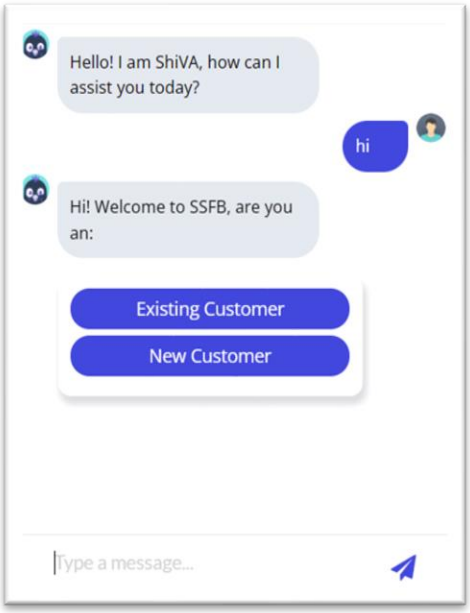


Fig 18 Flow for existing customers

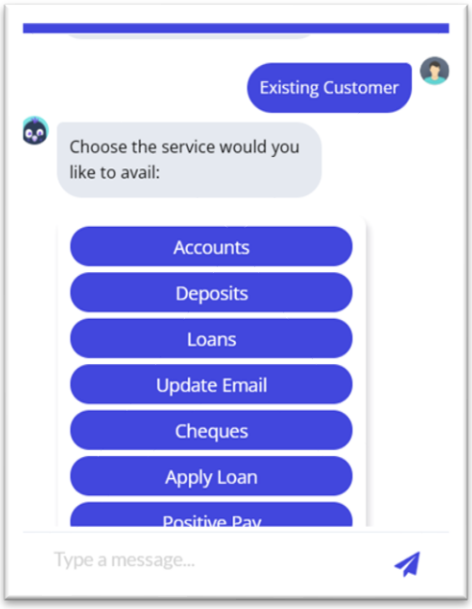


Fig 19 Flow for existing customers

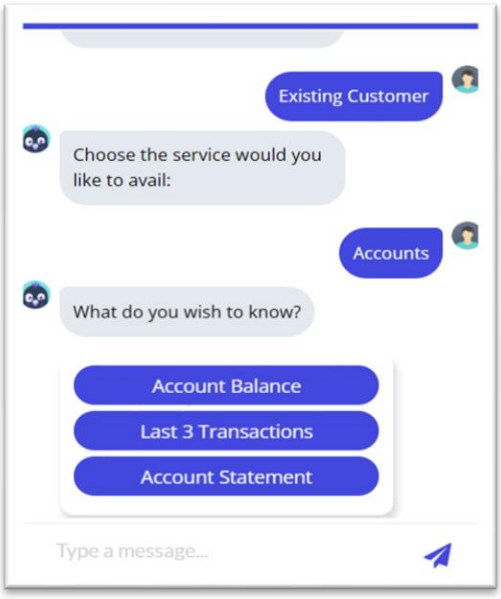


Fig 20 Flow for existing customers

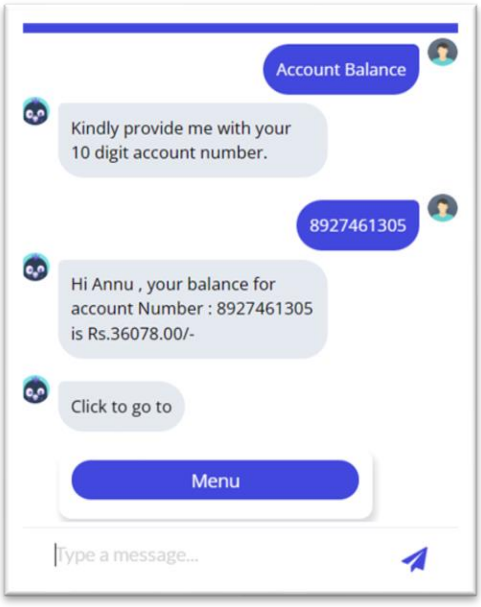


Fig 21 Flow for existing customers

Fig 18,19,20,21 Shows the flow of conversation for an existing customer in a bank where we have put in an authentication layer through the account number that is being asked from the customer and based on its verification the asked query is being displayed after it hits the API dataset and fetches back the particular user details.

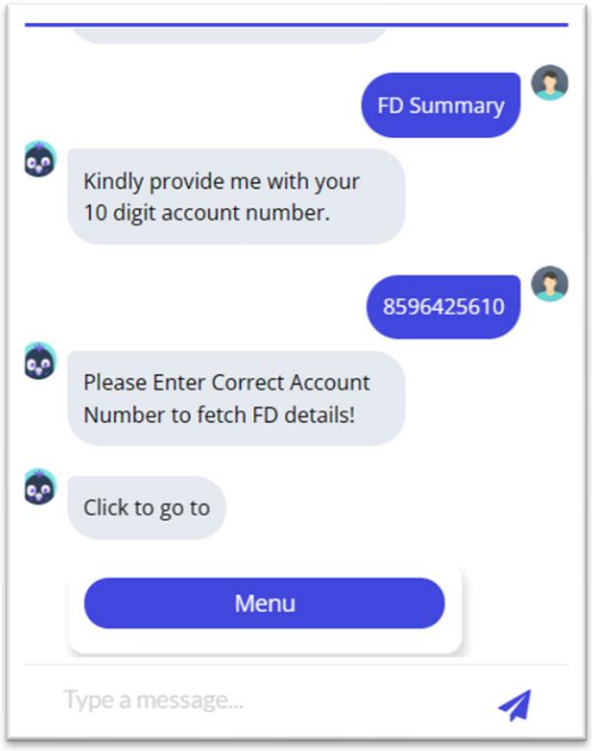


Fig 22 Fall Back Mechanism in case of wrong or invalid Account number

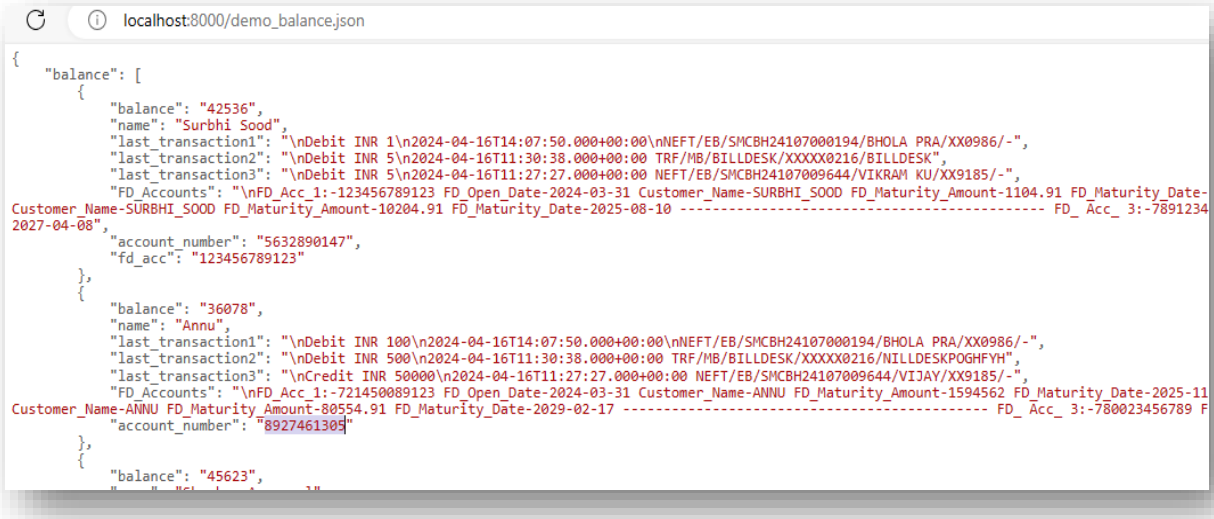


Fig 23 Locally hosted file from where the bot is fetching the data to be displayed as a response for custom actions.

5.2 COMPARISON WITH EXISTING SOLUTION

1. Strengths of Rasa Banking Bot
    - Customization and Flexibility: Rasa provides a high level of customization, allowing developers to tailor the chatbot's behaviour precisely to the banking domain's requirements.
    - Open Source: Rasa is an open-source framework, enabling transparency and flexibility in the development process without vendor lock-in.
    - Privacy and Security: The ability to deploy the bot without external API linkage can enhance data privacy and security, especially for organizations with stringent data protection policies.
    - Offline Mode: Rasa can operate in environments with limited or no internet connectivity, ensuring continuous service availability even in offline scenarios.
2. Limitations:
    - Limited Pre-Trained Models: Rasa requires substantial effort in creating and fine-tuning training data, which might be more resource-intensive compared to solutions with pre-trained models.
    - Dependency on Training Data Quality: The effectiveness of the bot heavily relies on the quality and diversity of the training data, and managing training data can be challenging.
    - Manual Integration: Without external API linkage, integrating with backend systems, databases, or third-party services may require more manual effort.
3. Comparison with Existing Solutions: Compared to Pre-Built Solutions (e.g., Banking Chatbots from Major Providers):
    - Advantages of Rasa:
        - Greater customization and adaptability to specific banking requirements.
        - Open-source nature allows complete control over the bot's development and deployment.

- Disadvantages of Rasa:
    - Requires more manual effort for training data creation and model tuning.
    - Integration with external systems may be more manual without out-of-the-box connectors.
    -

4. Compared to Rule-Based Systems:
    - Advantages of Rasa:
        - Natural language understanding capabilities surpass traditional rule-based systems.
        - Better adaptability to diverse user inputs.
    - Disadvantages of Rasa:
        - Training data requirements might be more extensive.
        - Initial development effort could be higher compared to rule-based approaches.

A Rasa banking bot provides distinct benefits, especially in terms of customisation and privacy. Nevertheless, it necessitates meticulous deliberation of the compromises involved, such as the requirement for extensive training data and labor-intensive integration endeavors. The selection between Rasa and current solutions is contingent upon the specific needs, availability of resources, and the desired degree of control over the development and deployment of the chatbot.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The Natural Language Chatbot developed using Rasa represents a significant advancement in conversational AI, offering valuable insights into the capabilities and limitations of contemporary chatbot frameworks.

- Key findings from the project include:

    1. Effective Intent Recognition and Entity Extraction: Rasa demonstrated robust performance in intent recognition and entity extraction, crucial for understanding user queries accurately.

    2. Contextual Dialogue Management: The chatbot successfully handled contextual conversations, showcasing Rasa's ability to manage dialogue flow and maintain context across multiple turns.

    3. Flexible Customization and Training: The project highlighted Rasa's flexibility, enabling easy customization of the chatbot's responses and efficient retraining to adapt to evolving user needs.

    4. Integration Capabilities: The chatbot seamlessly integrated with external systems and APIs, showcasing Rasa's versatility in connecting with diverse data sources to enrich responses.

    5. User Engagement and Satisfaction: User testing indicated positive feedback on the chatbot's natural language understanding and responsiveness, contributing to enhanced user engagement and satisfaction.

- Limitations:

    1. Data Dependency: The performance of the chatbot heavily relies on the quality and diversity of the training data. Limited datasets may result in gaps in understanding complex user queries.

    2. Handling Ambiguity: Despite advancements, the chatbot may struggle with ambiguous queries, highlighting a common challenge in natural language processing.

    3. Training Resource Intensity: Training and retraining the chatbot can be resource-intensive, especially as the dataset and complexity of conversations grow.

- Contributions to the Field:

    1. Open-Source Conversational AI: The project contributes to the open-source conversational AI community by showcasing the practical implementation and effectiveness of Rasa for developing sophisticated chatbots.

    2. Context-Aware Dialogue Systems: The emphasis on contextual dialogue management adds to the understanding of how chatbots can evolve beyond single-turn interactions, catering t o more dynamic and complex user conversations.

- Integration Best Practices: The project provides insights into best practices for integrating chatbots with external systems, a critical aspect for real-world applications where access to external data is vital.

6.2 FUTURE SCOPE

The future scope for a Rasa chatbot in the banking domain without external API linkage holds various opportunities for improvement, expansion, and enhancement. Here are several areas to consider for the future development of the chatbot:

1. Enhanced Natural Language Understanding (NLU): Invest in advanced NLU models, potentially incorporating state-of-the-art language models like GPT-4 or BERT, to improve the chatbot's understanding of user queries, intents, and entities.

2. Continuous Learning and Adaptation: Implement mechanisms for the chatbot to learn from real user interactions and continuously update its knowledge base. Leverage user feedback for model improvement and adaptability.

3. Multilingual Support: Extend the chatbot's capabilities to support multiple languages, catering to a diverse user base and expanding its reach to international customers.

4. Richer Conversational Experiences: Develop more sophisticated dialogue management strategies to enable the chatbot to engage in nuanced and context-aware conversations. Implement features like memory and context retention for more coherent interactions.

5. Improved Fallback Mechanism: Enhance the fallback mechanism to handle a wider range of out-of-scope queries more intelligently. Provide users with helpful suggestions or guide them back to relevant banking tasks seamlessly.
6. Integration with External Databases: Explore opportunities for secure integration with external databases, even without exposing APIs directly. This could involve implementing secure data connectors or exploring encrypted data retrieval mechanisms.
7. Voice and Speech Recognition: Integrate voice and speech recognition capabilities to allow users to interact with the chatbot using natural language processing in spoken form, providing a more accessible and convenient interface.

8. Advanced Security Measures: Strengthen security features by exploring biometric authentication, two-factor authentication, or other advanced security measures to ensure the protection of sensitive user information.
9. Personalization and User Profiles: Implement user profiles and personalization features to tailor the chatbot's responses and recommendations based on individual user preferences, transaction history, and behavior.
10. Regulatory Compliance: Stay abreast of evolving banking regulations and compliance standards. Update the chatbot to ensure it aligns with the latest legal and security requirements in the banking sector.

By focusing on these future scopes, a Rasa chatbot in the banking domain can evolve to meet the changing needs of users, offer more sophisticated and personalized services, and stay competitive in the dynamic landscape of conversational AI.

REFERENCES

[1] Venkatesham Boddula, CHATBOT USING RASA, International Research Journal of Modernization in Engineering Technology and Science Volume:04/Issue:08/August-2022

[2] Supreetha H V, Sandhya S , Implementation of an Educational Chatbot using Rasa Framework, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075 (Online), Volume-11 Issue-9, August 2022

[3] Kanchan B Malusare, RASA Chatbot Using AI, International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 2, Issue 1, April 2022

[4] D. Y. Patil, College Enquiry CHATBOT using RASA, International Journal of Scientific Research in Computer Science, Engineering and Information Technology ISSN : 2456-3307 (www.ijsrcseit.com) Volume 8, Issue 3, May-June-2021

[5] Wistiani Astuti, Predicting Frequently Asked Questions (FAQs) on the COVID-19 Chatbot using the DIET Classifier, East Indonesia Conference on Computer and Information Technology (EIConCIT) April 09-11. 2021, ISTTS Surabaya, Indonesia

[6] Kanakamedala Deepika, Jollity Chatbot- A contextual AI Assistant, International Conference on Smart Systems and Inventive Technology (ICSSIT 2020) IEEE Xplore

[7] Randil Pushpananda, The impact of using pre-trained word embeddings in Sinhala chatbots, 2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)

[8] Mayuresh Virkar, Humanizing the Chatbot with Semantics based Natural Language Generation, 2019 International Conference on Intelligent Computing and Control Systems (ICCS)

[9] Nuobei Shi, Language Chatbot–The Design and Implementation of English Language Transfer Learning Agent Apps, 2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)

[10] Soufyane Ayanouz, A Smart Chatbot Architecture based NLP and Machine learning for health care assistance, 2020 Association for Computing Machinery. ACM

[11] Rakesh Kumar Sharma, An Analytical Study and Review of open Source Chatbot framework, RASA , International Journal of Engineering Research & Technology (IJERT) Vol. 9 Issue 06, June-2020

[12] Tarun Lalwani, Shashank Bhalotia, International Journal of Innovative Research In Computer Science & Technology (IJIRCST) Volume-6, Issue-3, May-2018

[13] E. Kasthuri and S. Balaji, "A Chatbot for Changing Lifestyle in Education," Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV),

[14] Smutny, P., & Schreiberova , P. “Chatbots for learning: A review of educational chatbots for the Facebook Messenger,” Computers & Education, 2020

[15] M. Rakhra et al., "E-Commerce Assistance with a Smart Chatbot using Artificial Intelligence," 2nd International Conference on Intelligent Engineering and Management (ICIEM), 2021

[16] N. Shi, Q. Zeng and R. Lee, "Language Chatbot–The Design and Implementation of English Language Transfer Learning Agent Apps," IEEE 3rd International Conference on Automation, Electronics And Electrical Engineering (AUTEEE) , 2020

[17] N. N. Khin and K. M. Soe, "Question Answering based University Chatbot using Sequence to Sequence Model," 23rd Conference of the Oriental COCOSDA International Committee
(O-COCOSDA), 2020

[18] S. García-Méndez, F. De Arriba-Pérez, F. J. González-Castaño, J. A. Regueiro-Janeiro and F. Gil-Castiñeira, "Entertainment Chatbot for the Digital Inclusion of Elderly People Without Abstraction Capabilities ," in IEEE Access, vol. 9, 2021, pp. 75878-75891.

[19] M. N. Kumar, P. C. L. Chandar, A. V. Prasad and K. Sumangali, "Android based educational Chatbot for visually impaired people," IEEE International Conference on Computational Intelligence And Computing Research (ICCIC), 2016

[20] G. Daniel and J. Cabot, "The Software Challenges of Building Smart Chatbots, "IEEE /ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) , 2021

[21] E. H. Wu, C. Lin, Y. Ou, C. Liu, W. Wang and C. Chao, "Advantages and Constraints Of a Hybrid Model K-12 E-Learning Assistant Chatbot," in IEEE Access, vol. 8, 2020

[22] L. E. Chen, S. Y. Cheng and J. -S. Heh, "Chatbot : A Question Answering System for Student, " International Conference on Advanced Learning Technologies (ICA LT), 2021

[23] RASA Architecture Overview, RASA Docs, https:// rasa. com /docs/rasa /arch - overview/

[24] A. Pal, P. Parhi and M. Aggarwal, "An improved content based collaborative filtering algorithm for movie recommendations," 2017 Tenth International Conference on Contemporary Computing (IC3), 2017

[25] B. Patel, P. Desai and U. Panchal, "Methods of recommender system: A review ,"2017 International Conference on Innovations In Information, Embedded and Communication Systems (ICIIE CS), 2017