



Intelligent Chat Bot for the Banking Sector

**A Dissertation Submitted for the Degree of
Master of Computer Science**

W. P. B. G. Warnakulasooriya
University of Colombo School of Computing
2021




DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Student Name: W.P.B.G. Warnakulasooriya

Registration Number: 2018MCS095

Index Number: 18440954

 27.11.2021

Signature of the Student & Date

This is to certify that this thesis is based on the work of Mr. W.P.B.G. Warnakulasooriya under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by,

Supervisor Name: Dr.A. Ruvan Weerasinghe

Signature of the Supervisor & Date

This thesis is especially dedicated to my beloved parents and all the family members who were there with me in all my ups and downs; And for all the students who are willing to implement new features in the same research category

ACKNOWLEDGEMENTS

With the generous support and assistance of many people, this thesis became a reality. For all of them I am expressing my true feelings here with the help of few words.

Foremost, I would like to express my sincere gratitude to my supervisor Dr. A. Ruvan Weerasinghe for his unwavering support of my master's research, as well as his patience, encouragement, enthusiasm, and immense knowledge. His advice was extremely valuable during the research and writing of this thesis. I could not imagine having a better advisor and mentor for my master's study.

I would like to thank Dr. B. H. Randil Pushpananda who's the MCS project coordinator who has given the advice and guidelines from the first step and until now. Moreover, all the other senior lectures and the assistant lectures who have helped me even with a single word should be heartily mentioned.

All the other academic staff members, academic and non-academic staff who engaged, guided and encouraged me during the course of Master Computer Science program of University of Colombo School of Computing.

All the call center and branch staff of Bank of Ceylon Head office within the time of data collection. And all the participants who are involve with the evaluation process which has done to ensure the level of success the final outcome.

Within this hard time period, all my family members, friends and colleagues of the MCS program have given a huge support to bring down the stress level being humble and kind. So, I would like to thank them all from the bottom of my heart.

Eventually, all personnel mentioned above and not mentioned here by names, I would like to express my honest and heartier gratitude with these few simple and short words.

W P B G Warnakulasooriya.

ABSTRACT

Customer Service is the most important part of the growing business. It is nothing but have the one-on-one communication between the consumer and the organization while introducing their products. Most of the businesses believe making connections with customers at their best level will guide the customers to bind with the same organization. Since most of the customers are not aware about how to proceed with the self-engagement, more manpower and the customer relationship management has become a must. Due to that reason a little mistake can hamper the organization name if the customer service went wrong.

Since this is the digital era, communication platforms with novel concepts of machine learning and artificial intelligence are rapidly growing. With advanced digital technologies related to machine learning and artificial intelligence, managing customer service has become easy, reliable and cost effective rather than applying human resources.

As a solution an AI assistant can be proposed: which can be used to avoid the disadvantages caused by human engagement as customer care employees. The AI assistant is nothing but a chatbot which helps the customers to inquire about the issues they are facing without having human interaction. Within the whole scenario the organization is considered as a bank. The importance of having an AI assistant is because organizations need to deliver more convenient communication mechanism that support 24-hour customer service, consistent answering, diminish the customer waiting time and instant communication. To achieve those goals organization able to develop their own Artificial Intelligence based assistant to cater their customers. AI assistant will enhance the productivity of bank staff, save overall costs, and maximize the customer engagement.

The banking AI assistant is responsible for responding to the customer's needs. To fulfill this requirement, first of all the customers should send their queries to the AI assistant. Once it is done, the AI assistant will identify the user's requirement and respond accordingly for the trained queries. Moreover, it will identify the misspelled user queries and respond to them within an approximate level. Other than that, AI assistant has the capability of providing answers which has lower possibility of being accurate with the graceful fallbacks according to the user queries. Even for the out of scope queries it responds intuitively.

TABLE OF CONTENT

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF FIGURES	vii
CHAPTER 1: INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Statement of the problem	1
1.3 Research Aims and Objectives	1
1.3.1 Aim	2
1.3.2 Objectives	2
1.4 Scope.....	2
1.5 Structure of the Thesis	3
CHAPTER 2: LITERATURE REVIEW.....	4
2.1 History	4
2.1.1 ELIZA.....	4
2.1.2 ALICE	4
2.2 Related work	5
2.2.1 Banking Chatbot on Turkish	5
2.2.2 AI Hotel Booking	5
2.2.3 AllenNLP.....	5
2.2.4 Bidirectional RNN and Attention model	6
2.2.5 Emotionally Realistic Chatbot Framework	6
2.3 Chatbot Frameworks.....	6
2.3.1 Google’s Dialogflow	6
2.3.2 Microsoft LUIS	7
2.3.3 Amazon’s Lex	7
2.3.4 IBM Watson	7
2.3.5 Rasa	8
CHAPTER 3: METHODOLOGY.....	9
3.1 Chatbot Architecture.....	9
3.1.1 Connector module	10

3.1.2	Natural Language Understanding unit.....	10
3.1.3	Dialog Management unit	12
3.3	Chatbot Design and Implementation	14
3.3.1	Data collecting Approach	14
3.3.2	NLU unit Design	15
3.3.3	Training Stories Design	19
3.3.4	Domain Design.....	21
CHAPTER 4: RESULTS AND EVALUATION.....		30
4.1	Results.....	30
4.2	Evaluation	34
4.2.1	Evaluation approach	34
4.2.2	Participant.....	35
4.2.3	Evaluation on Prototype 01	35
4.2.4	Evaluation on Prototype 02	38
4.3	NLU Testing Result	40
4.4	Summary.....	41
CHAPTER 5: CONCLUSION AND FUTURE WORK.....		42
5.1	Conclusion	42
5.2	Limitation.....	43
5.3	Future Works	43
APPENDICES.....		I
6.1	Appendices 1 – Questionnaire	I
REFERENCES.....		III

LIST OF FIGURES

Figure 1 : Architecture.....	9
Figure 2 : Tokenizer	11
Figure 3 : Count Vectorizer	12
Figure 4 : NLU dataset “Search account”	16
Figure 5 : NLU dataset “greet”	17
Figure 6 : NLU dataset “out-of-scope”.....	18
Figure 7 : NLU dataset “synonym”	19
Figure 8 : Training stories “search products.....	20
Figure 9 : Training stories “suggesting account”	21
Figure 10 : Domain “responses”.....	22
Figure 11 : Domain “forms”	23
Figure 12 : Custom action “account type search”	24
Figure 13 : Custom action “database connection”.....	25
Figure 14 : Custom action “intent description mapping”	25
Figure 15 : Config “pipeline and policies”	26
Figure 16 : Rules	27
Figure 17 : VM instance in Google Cloud Platform	28
Figure 18 : Conversation History in Rasa X Platform	29
Figure 19 : Interface “slack messaging platform”	30
Figure 20 : Interface “web interface”	31
Figure 21 : Respond “account search”	31
Figure 22 : Respond “suggesting account I”	32
Figure 23 : Respond “suggesting account II”	32
Figure 24 : Responds “2 stage fallback”	33
Figure 25 : Responds “out-of-scope”	33
Figure 26 : Responds “synonym”	34
Figure 27 : Responds “synonym Misspell”	34
Figure 28 : Overall chatbot evaluation – prototype 01	36
Figure 29 : Evaluation results category wise - prototype 01	37
Figure 30 : Overall chatbot evaluation – prototype 02.....	39
Figure 31 : Evaluation results category wise - prototype 02	40
Figure 32 : Intent confusion matrix	40
Figure 33 : Intent histogram	41

CHAPTER 1: INTRODUCTION

Banking is a major part of our day today life. Building up savings, drawing salaries, paying utility bills, borrowing loans all involve transactions with the Bank. All the businesses relying on the banking sector since they handle their financial transactions such as day to day deposits/ withdrawals, investments, business expansions plan to name a few.

But if we have any banking related queries, we have to visit the bank or contact customer care. This consumes a considerable amount of time and effort from both parties, especially for the bank. Huge amount of phone calls regarding customer inquiries are attended by the bank staff daily regarding Loans, Fixed Deposits, Exchange rates, Leasing, Pawning etc. lack of staff and heavy workload per employee sometimes result in unattended customer inquiries and delays in assisting the customer. Therefore, it will be more convenient if customers have a proper way of communicating with the bank via online process and get the feedback within a short period of time.

1.1 Motivation

To overcome this particular issue, a chat bot which is capable of assisting customers with accurate information instantly will be very useful and convenient for both customers and the banking sector. Customers can directly communicate with the chatbot, and they can receive answers to their bank related queries. Also, the bank can reduce their customer query phone calls by using a chat bot which they can assist customers with accurate and updated information.

1.2 Statement of the problem

Problem is that find any financial information regarding Commercial Banks are not very pleasant at the moment. For an example, if we want any information about a loan scheme either we must physically visit the bank or contact customer care. Therefore, developing a proper chatbot that can Process Natural Language (NLP), find highest convenient answer to user's question and answer more human friendly way to the user's question is the final goal of this thesis.

1.3 Research Aims and Objectives

Aim and objective of this thesis is to implement and evaluate AI based chatbot that can process Natural Language in the banking domain.

1.3.1 Aim

Aim of this project is to implement an Artificial Intelligence banking assistant to answer banking related customer queries and diminish customer inquiry calls to the bank staff. Without waiting in a queue, customers can have answers to their queries or get assistance and nevertheless bank staff can use their valuable time to enhance their productivity.

1.3.2 Objectives

Implementing and evaluating user friendly banking chatbot is the objective of this thesis. And to achieve this objective,

- **Round the clock support-** Increasingly digital always-on consumers are expected 24-hour customer service. Customers are more likely to be attracted and retained if a chatbot is integrated into the bank's website.
- **Enhanced productivity of bank personnel-** Banking chatbot can make bank personnel more productive by allowing them the freedom to focus on more complex problems instead of being stuck with basic customer queries.
- **More convenient mode of communication** – Younger generation prefer instant messaging and it is faster than waiting in a queue to get assisted by a staff member.
- **Consistent answers-** Chatbot maintain consistency in answering user queries and this ensures value to customer conversation. Consistent answers will always improve customer experience with the bank.

1.4 Scope

Scope of this research is to develop a chatbot that can serve customers via bank website and Slack. When chatbot built into the bank website have a competitive advantage and more importantly can have attention from interested people. And this chatbot has to answer Current, Savings and Fixed Deposit related queries. Most of the time customers have queries related to the bank products. Therefore, chatbot has to assist customers on product related queries. These customer queries need to receive via textual format as well as responds need to show as textual format. Chatbot has to simulate human conversation. To give better experience to the customer, chatbot needs to maintain smooth conversation flow and maintain polite responses when answering doubtful queries. Answers have to be highly relevant to the user's queries.

1.5 Structure of the Thesis

Chapter 2 presents the academic and subject background of chatbot and Literature review. Methodology of the thesis presented in Chapter 3. Chapter 4 describes the Evaluation of the chatbot and result. And finally, Chapter 5 presents the Conclusion of the study, including limitations and future work opportunities.

CHAPTER 2: LITERATURE REVIEW

This chapter provides essential background information referring to chatbot projects. Most importantly in this chapter discover the history of chatbot, recent chatbot projects and chatbot frameworks available.

2.1 History

2.1.1 ELIZA

In 1964 Joseph Weizenbaum developed a Natural Language Processing Computer program called ELIZA at MIT Artificial Intelligence Laboratory (Weizenbaum, 1966). ELIZA was one of the world's first chat bot programs. To simulate conversation, ELIZA employs pattern matching and replacement techniques. However, there is no built-in framework for contextualizing events in this program. ELIZA was built in MAD-SLIP, which allows to analyze user inputs and carry on a conversation based on the constrains given in the SCRIPTS. ELIZA uses keyword matching, which implies that the program will look for keywords that match the input. If matching keywords are identified, the system will generate a suitable response based on the criteria specified for this keyword, and if not, a related remark will be recalled. As a result, ELIZA does not always grasp the problem or the user's input; instead, ELIZA just matches the user's input with her regular responses.

2.1.2 ALICE

Artificial Linguistic Internet Computer Entity (ALICE) is a Natural Language Processing chatbot which can proceed conversations by responding as natural as possible("A Closer Look at Chatbot ALICE," n.d.; "AliceBot," n.d.) . Richard Wallace developed ALICE in 1995. ALICE can use a fallback system using linguistic deflection. ALICE stores its knowledge about English conversation patterns in AIML files (Artificial Intelligence Markup Language). AIML files contain data objects called AIML objects, that can divide as topics and categories. The topics have a name attribute and a set of categories related to this specific topic, while categories refer to the basic unit of knowledge in AIML. Each category has a pattern and a template as well as a guideline for aligning the user's input to the desired result.

2.2 Related work

2.2.1 Banking Chatbot on Turkish

Yapi kredi Technology's developed a Hybrid Approach to question answering for a banking chatbot on Turkish (Dünder et al., 2018). Specialty of this research is they have proposed hybrid key word- word embedding based question answering method. This will calculate similarity between two questions and similar words are stored in the database using clustering. Word embeddings are used to get semantic similarities between synonyms. Using synonymous they can identify the most relevant words in the bag of words. Queries and dataset are matched, and performance has been increased compared to methods that do not utilize keywords. But when handling morphologically rich languages need to expand the keyword database farther.

2.2.2 AI Hotel Booking

Researchers at the University of Toronto developed a Real-world conversational AI system for hotel booking (Li et al., 2019). They have used a frame-based dialogue management system that integrates with NLP. Drawback is that more complex queries need to be handled by human. And also, deep learning models are memory intensive and it's important to share memory across different models when the system expands in the future.

2.2.3 AllenNLP

Allen Institute for Artificial Intelligence developed a Deep Semantic Natural Language Platform (Gardner et al., 2018). AllenNLP platform that addresses issues with easy-to-use command-line tools, declarative configuration-driven experiments, and modular NLP abstractions. This will allow researchers to focus on the high-level summary of their models rather than the details.

2.2.4 Bidirectional RNN and Attention model

Bajaj Institute of Technology developed an Intelligent chatbot using deep learning with Bidirectional RNN and attention model (Dhyani and Kumar, 2020). Bidirectional Recurrent Neural Network contain attention layers is used, so that input sentences with large number of tokens can be replied with more appropriate conversations. This is an open domain chatbot that can develop for a particular domain by training with the domain knowledge.

2.2.5 Emotionally Realistic Chatbot Framework

Researchers of Bina Nusantara University designed an Emotionally Realistic chatbot framework to Enhance Its believability with AIML and Information states (“Designing an Emotionally Realistic Chatbot Framework to Enhance Its Believability with AIML and Information States,” 2019). Main focus of the research is Understanding a natural conversation and replying and keeping the conversation flowing naturally.

2.3 Chatbot Frameworks

2.3.1 Google’s Dialogflow

Dialogflow is a google owned natural language understanding AI platform that makes conversational user interface into devices in many different contexts (“Dialogflow Documentation,” n.d.). Such as mobile apps, bots, interactive voice response systems. Dialogflow that uses Natural language processing NLP, actions on Google and Google cloud platform that expose

artificial intelligence and machine learning methods such as natural language methods such as natural language understanding (Chinnapa Reddy Kanakanti and Sabitha R., 2020). Dialogflow provide multiple types of input from users including voice and text inputs and also Dialogflow can respond them in either through text or with synthetic speech. Dialogflow introduce two different services, Dialogflow CX provide an advanced agent and Dialogflow ES provide the slandered agent. Dialogflow CX capable to customize and improve performance and accuracy of the response.

2.3.2 Microsoft LUIS

Microsoft Language Understanding Intelligent Service (LUIS) is a cloud based conversational AI service that provides custom machine learning to a user's conversational, natural language text to predict overall meaning, and pull out relevant detailed information (aahill, n.d.). LUIS provides enterprise ready custom models that continuously improve conversational ability. With LUIS, developers without machine learning expertise can quickly build and use language understanding models specific to their tasks (Williams et al., 2015). but there are some limitations with the customization.

2.3.3 Amazon's Lex

Lex is a conversational interface building service introduced by Amazon, and it will integrate with any application and be able to use voice and text as inputs (What Is Amazon Lex? - Amazon Lex). Lex provides the advanced deep learning functionalities of Automatic Speech Recognition (ASR) for converting speech to text and Natural Language Understanding (NLU) to recognize the intent of the text to enable to build applications with highly engaging user experiences and lifelike conversational interactions. **Developing of a voice chatbot for payment using Lex service with Eyowo as the Payment platform** (Samuel et al., 2020). The voice chatbot system will provide an alternative means of engaging in financial transactions just by speaking, thereby fostering innovations in the technology industry, this will be done by using Amazon Lex and Lambda function, alongside the use of a payment platform called Eyowo, in resolving all financial matters.

2.3.4 IBM Watson

Watson is a question answering computer system capable of answering questions posed in natural language, that developed by IBM's DeepQA project by a research team (IBM, 2020). Watson Assistant provides users with consistent and accurate answers across any device, channel, or application. Using Artificial Intelligence, Watson Assistant learns from user conversations, and improving capabilities. **Practicing value innovation through artificial intelligence** (Russo-Spena et al., 2019). This article focuses on the value innovation prompted by artificial intelligence. By shifting the attention from innovation as a new outcome to innovating as something that people do to co-create value. Detecting how multiple actors

connect, interact, learn and discover new ways to do things, serve others better and co-create value through AI.

2.3.5 Rasa

Rasa is an open source framework for developing AI powered, industrial grade chatbots (“Open source conversational AI,” 2020). **An analytical study and review of open source chatbot framework** (Sharma, 2020). Rasa can interact with both voice-based and text-based conversations also able to develop chatbots to different platforms and channels. Speciality of the platform is providing infrastructure and tools necessary for high performing, resilient, proprietary contextual assistants that work. Rasa’s NLU provides developers with the technology to understand messages, determine intent, and capture key contextual information. Framework supports multiple languages, single and multiple intents also both pre trained and custom entities with more control. Rasa’s Dialogue management is able to hold meaningful conversations with user’s multi step conversations that remember context and integrate business logic. And also, it learns interactively from real conversations. Rasa focuses on flexible architecture not on straight out of the box software because it is necessary for great conversational AI.

CHAPTER 3: METHODOLOGY

In this section, we will cover some general principles, methods and approaches that were used throughout the entire Rasa based banking assistant implementation. Data collecting approach, design conversation and creating NLU data, training NLU model, dialogue management model and training the NLU model using real user conversations are the main areas that are discussing through this chapter.

3.1 Chatbot Architecture

Banking Chatbot will develop using the RASA framework. Architecture will describe the process under the framework and methodology that was used to implement the chatbot. Chatbot architecture can be describe as Figure 1.

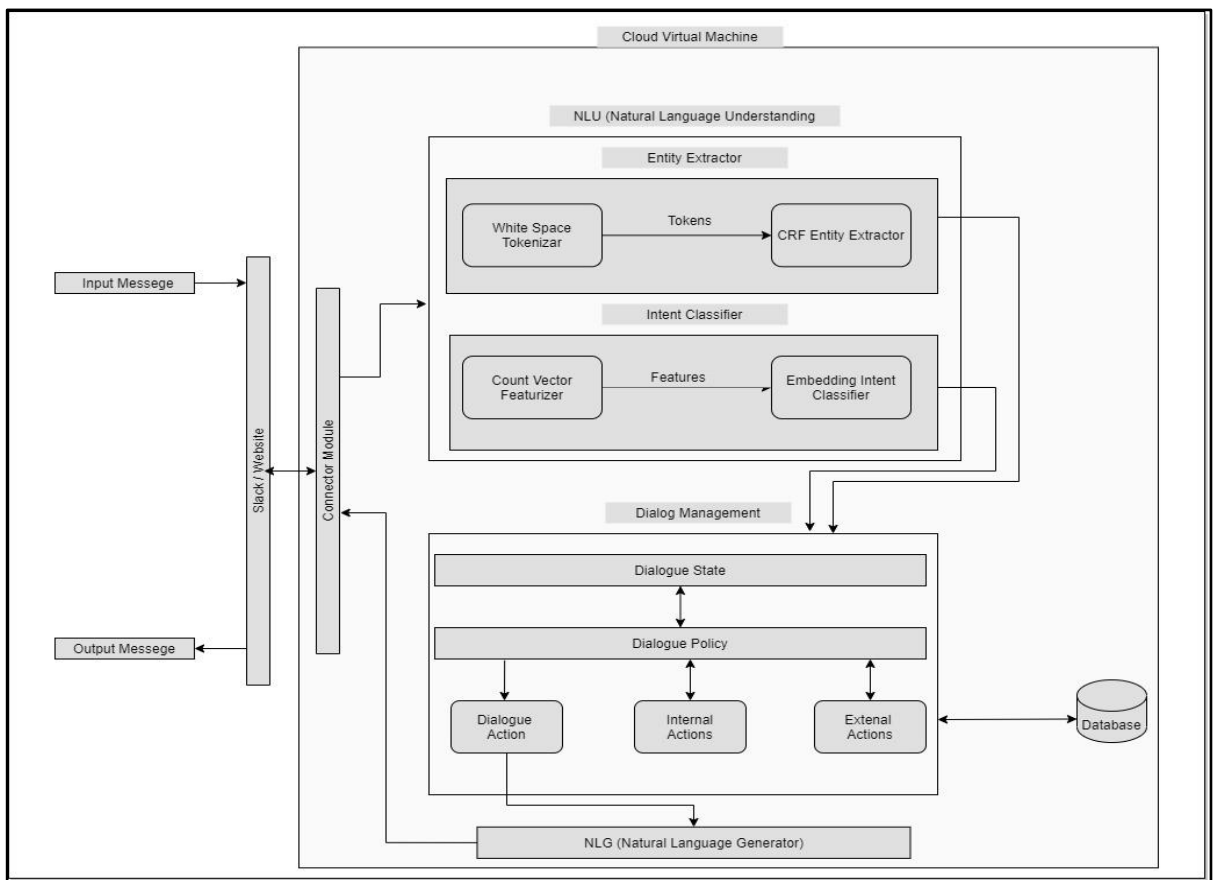


Figure 1 : Architecture

3.1.1 Connector module

Customer queries can be sent via Slack messaging platform and also customers can use chatbot using the bank website. Both the interfaces (slack and the website) are connected to the connector module. This is a two-way connection slack, and the website can send messages through the connector plus both interfaces can receive messages through the connector module.

3.1.2 Natural Language Understanding unit

The connector module will send customer query to the Natural Language Understanding (NLU) unit. NLU unit include two processes, they are Entity Extractor and Intent Classifier. Intent of the customer query and the entities in the query will be carried out using those Entity extractor and Intent classifier.

3.1.2.1 Entity Extractor

First thing of this NLU unit is to tokenize user queries to understand a meaningful pattern of the user's input. To understand the meaning of the sentence, tokenizer will split the full sentence into smaller unites. We are developing a domain specific AI assistant therefore, the best option for the tokenizer is **WhitespaceTokenizer**. This tokenizer will look for a white space of the sentence and create a token for every white space separated character sequence.

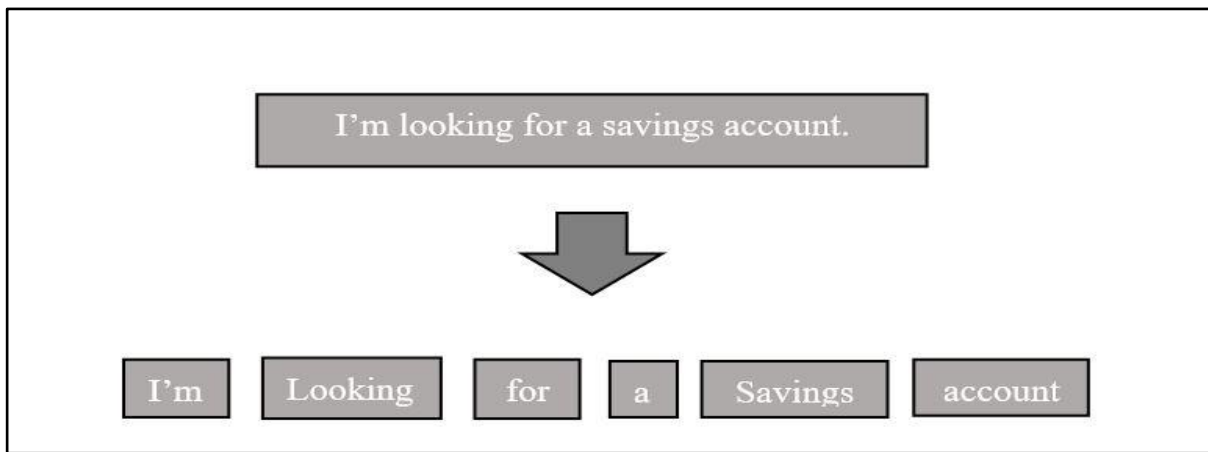


Figure 2 : Tokenizer

These tokens created by tokenizer usually used by quite a few subsequence components in the pipeline. First thing is Entity extractor, this will use tokens to extract the entities from the sentence. Most robust entity extractor of the Rasa Framework is **CRFEntityExtractor** which defines the model called Conditional Random Field which learns to identify which words are entities in a sentence and what kind of entities they are by observing the sequence of tokens. Entity extractor is picking a target word and looks at the surrounding words and extract features of the words like word starts with uppercase or lowercase, prefix or suffix, word, or a digit. This output will show which words are entities and what are their labels, and how confidence the model was making those predictions.

3.1.2.2 Intent Classifier

Intent classification is the next step of the NLU unit. For the intent classification initially need two main components Featurizer and actual model. Featurizer create new features using tokens which can be used by an intent classification model to learn patterns and make the predictions. **CountVectorsFeaturizer** is a great choice for a domain specific AI assistant. This featurizer creates bag of words representation of user message and label features using CountVectorizer. Featurizer counts how often distinct words of training data appear in a message and provides that as input for the intent classification model.

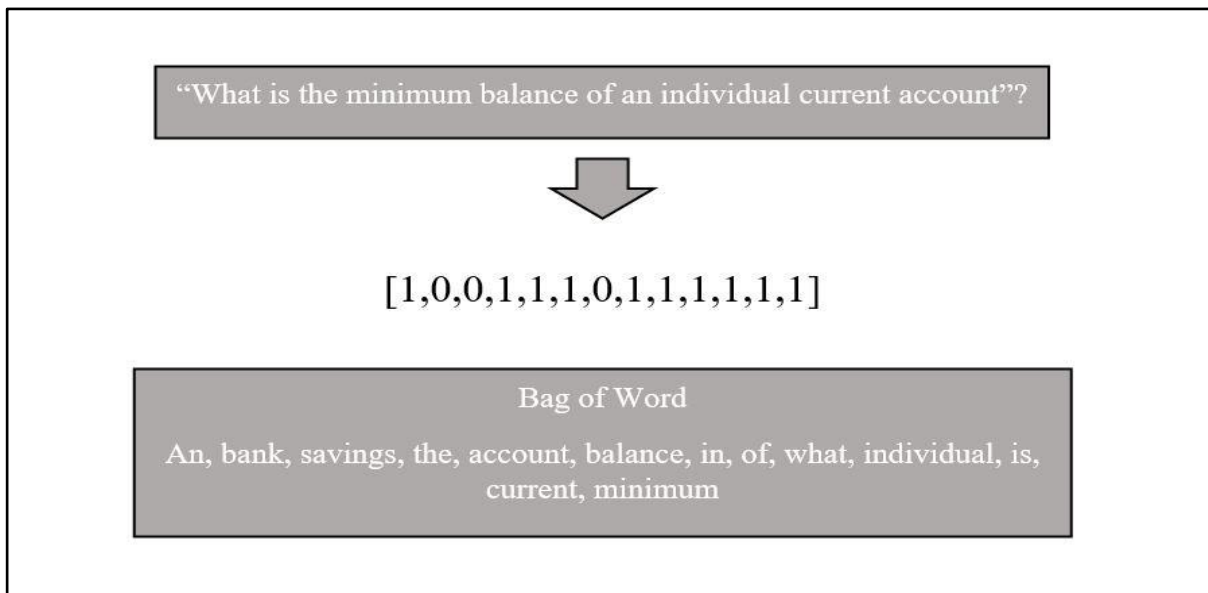


Figure 3 : Count Vectorizer

Once the features are extracted, they are good to be used to train the intent classification model. **EmbeddingIntentClassifier** is the choice of the intent classification model and when features extracted by the featurizer will be fed into the EmbeddingIntentClassifier and the intent classification results will be produced. The EmbeddingIntentClassifier embeds user inputs and intent labels into the same space. Embeddings are trained by maximizing similarity between them and most probable intent is producing. And also, it provides a ranking of the labels that did not win.

3.1.3 Dialog Management unit

Once the entities and intent are extracted that data will be fed into the dialog management unit. Dialog management unit includes three layers. They are, Dialog state, Dialog Policy and Actions.

3.1.3.1 Dialog Policy

When intents and entities are passed to the Dialog management unit, Dialog policies can models mimicking the conversations that they were trained on, to quite sophisticated machine learning models capable of predicting next action based on the history of the conversation, context, and the other important information. **Memorization policy** is mimicking the stories it was trained

on and depending on max_history parameter was set it tries to match the fragment of the current story with the stories provided in the training data file. In this Bank chatbot project we are setting max_history value as 4 and therefore, Memorization policy will only focus on the last four conversations and find the matching stories from the training data. When the memorization policy finds a matching fragment, it predicts the next action from the matched story with a confidence one.

We have to use **Keras Policy** to predict the next action. Keras policy uses the neural network implemented in a python deep learning library called keras. This policy will take into account details like “what is last action, what is the intent and what entities extract from the current user inputs, what slots are set at the moment” to predict the next action. In addition to this, the policy will consider the previous state of the dialogue. When we set max_history to four this policy will also consider only the last four states to predict the next action.

When we consider conversational AI, we have to face situations where the users ask for something that the assistant wasn't really designed to do, or the user can provide inputs that make NLU and dialogue manager models rather confusing. Because the assistant does not have enough knowledge to handle certain inputs. Training an AI assistant for all the situations is not possible therefore, enabling the assistant to acknowledge these situations and handle them gracefully is an important task. To fulfill this purpose, we have to use **Fallback policy**. This will allow us to set the thresholds for the NLU and dialogue management models. And if they are not found the training stories related to user input fallback action is predicted by the fallback policy. Using the fallback action assistant can respond politely like “sorry, I didn't get that”, “can you rephrase” to avoid such a situation. We can configure minimum confidence needed to accept the NLU and dialogue management models.

3.1.3.2 Actions and Natural Language Generator

Dialog policy will decide which action needs to be performed next and action will trigger the process. Actions can be Internal or External. Internal actions are simple answers that contain internal boundary. For example, if customer greet, chatbot needs to react in the same manner. These kinds of actions are internal. In other hand External actions are connected to the external components. Most of the time external actions are connected to external databases or API's. Using those external components action can provide better and sophisticated responses to the

customers. For example, if customer inquire “interest rate of the account”, external action can connect to the database and provide reasonable a response. After constructing the response, dialog action will send the result to the Natural Language Generator and the NLG unit sends the response to the connector module. Finally, the message will be displayed to users on slack or the website.

3.3 Chatbot Design and Implementation

3.3.1 Data collecting Approach

As the first step, we have to work with domain experts and leverage their knowledge of the customer base. And also, we can collect most common search queries related to saving, current and fixed deposit accounts using the bank website. This information would help us to define most common questions the potential audience may ask.

Bank front office staff and the call center staff handle the most customer queries. Initially we visited the call center and presented the project intent and objective. After providing sufficient details about the proposed Banking bot, we have collected the relevant user queries and related information. To gather sufficient user queries, we have use branch front office staff as well. Customers usually visit the branch and ask their accounts related questions form the Personal Banking Unit. Therefore, they have experience about customer queries and their needs. With the support of different departments, we were able to increase the quality of the most frequent user queries and data collection process.

During the data collecting phase, we have collected different user queries from different departments. After the collecting phase was finished, we had to sort the queries according to different intents and different business areas like account related, ATM related, Digital Banking related and general banking. After sorting the queries, we have selected only the queries which are related to the scope. In this research we are considering account related queries therefore, we have selected only savings, current and fixed deposit related queries and most frequent user queries.

3.3.2 NLU unit Design

After finalizing the queries which are related to the scope, we can design a Natural Language Understanding file. Using selected most common queries we can partition into different intents and extract entities related to the query. Natural Language Understanding is the most important task in the chatbot designing. Users can inquire about the same question in different ways. Therefore, training NLU file needs to group different queries that are related to similar intent. NLU needs to cover most of the different ways that users can ask queries. That will ensure the chatbot's understanding accuracy. When improving the NLU data set, increase the accuracy of the chatbot's understanding accuracy directly. In order to do that we have to categorize intents correctly and fill the intents with correct user queries in different ways. And that is important. This will directly be involving with develop more intelligent NLU model.

Next part is labeling the entities. Entities are usable information that are contained in the user queries. Extracting that information is crucial to clarify user requirements accurately. For example, customers can ask "what are the available accounts", in this situation there are no entities that contain the user query. Therefore, chatbot need to show all the account categories that are available in the bank. In the other hand if a customer enquire "what are the available savings accounts", this query includes an entity. Entity is the "savings account", when the chatbot extracts the entity precisely chatbot needs to show only available accounts related to the savings category. Therefore, identifying user given entities is an important task.


```

! nlu.yml x
data > ! nlu.yml > [ ] nlu > {} 38 > examples
479 - intent: search_account_types
480 - examples: |
481   - types of accounts
482   - [show me](search_account) types of accounts
483   - [show me](search_account) some types of accounts
484   - please, [tell me](search_account) about types of accounts
485   - [print info](search_account) about accounts
486   - [what are](search_account) the accounts available?
487   - accounts
488   - types of [savings](account_type) accounts
489   - please [tell me](search_account) about type of accounts
490   - [what are](search_account) the account types available
491   - [what are](search_account) the available account types
492   - [what are](search_account) the account types available
493   - [what are](search_account) the account types available
494 - intent: search_products
495 - examples: |
496   - [i need to open](facility_type) an account
497   - [i need to open](facility_type) a [savings](account_type) account
498   - [can i open](facility_type) a [current](account_type) account?
499   - [can i open](facility_type) an account?
500   - [suggest me](facility_type) an account
501   - [suggest me](facility_type) a [savings](account_type) account
502   - [help me to find](facility_type) an account
503   - [find me](facility_type) a [savings](account_type) account
504   - [find me](facility_type) a [current](account_type) account
505   - [find me](facility_type) a [fixed](account_type) deposit account
506   - [find me](facility_type) a [fixed](account_type) account
507   - [info about accounts](facility_type)

```

Figure 4 : NLU dataset “Search account”

Figure 4 shows the intent and entity labeling. Search_account_types and Search_products are two intents. And some queries can contain one or many entities and some queries may not contain any entities. For example, users can inquire simply “types of accounts”, this sentence does not contain any entity. “Types of **savings** accounts” this sentence contains one entity. “Savings” is an account_type entity. Furthermore, sentences can have more than one entity. “**I need to open a savings** account” in this sentence there are two entities encountered. “I need to open” is a facility_type and the other one is “savings”, that is an account_type entity.

When considering the above example, we can comprehend the benefit of extracting the entities in the user queries. Actual meaning of the customer query will not identify accurately when NLU data only considers intent classification of the sentence. “Types of savings accounts” intent of this sentence is Search_account_types. If the chatbot model only considers the intent, the chatbot either needs to request an account type from the user or the chatbot needs to respond with all the available account types in different categories.

Typically, users are not very pleasant with answering to bot’s queries. Therefore, reducing customer involvement is crucial to have better customer experience. In order to reduce the involvement of the user chatbot model able to recognize the meaning of the user query at once. Intent classification and the entity extraction will recognize the information that includes the

sentence and label them according to the training dataset. Entity extraction will raise the accuracy level of the natural language understanding process.

```
! nlu.yml x
data > ! nlu.yml > [ ] nlu > { } 1 > examples
3 - intent: greet
4   examples: |
5     - hey
6     - hello
7     - hi
8     - hello there
9     - good morning
10    - good evening
11    - moin
12    - hey there
13    - let's go
14    - hey dude
15    - goodmorning
16    - goodevening
17    - good afternoon
18 - intent: goodbye
19   examples: |
20     - goodbye
21     - cu
22     - good by
23     - see you later
24     - good night
25     - bye
26     - cee you later
27     - have a nice day
28     - see you around
29     - bye bye
30 - intent: affirm
31   examples: |
32     - yes
33     - y
34     - indeed
35     - of course
```

Figure 5 : NLU dataset “greet”

Addition to user queries chatbot able to understand greetings, goodbyes, thank, affirms, and denies responses. These greetings and other mentioned responses can be varying from user to user and time that conversation took place. In order to handle different user inputs, model needs to train using various different customer inputs. And the most important fact is users can provide misspell words. To handle those situations smoothly, chatbot model needs to train using NLU data with similar patterns.

Figure 5 shows some of the greet goodbye and affirm NLU data that we used in the chatbot implementation. As I mentioned in earlier users may post user queries with misspell or without even completing the sentence. And in addition to that, users will communicate using acronyms. Chatbot needs to distinguish that sort of acronyms, misspells words or incomplete words that contain the sentence. Chatbot model will gain that knowledge from the training NLU dataset and therefore, it is necessary to have a similar sort of dataset in the training NLU dataset and train the model accordingly.

```
! nlu.yml x
data > ! nlu.yml > [ ] nlu > { } 17 > examples
292 - intent: out_of_scope
293   examples: |
294     - what is the square root of 5
295     - what is the meaning of life.
296     - Fridge Isn't Running
297     - my tv isn't working
298     - I want a pizza
299     - my washing machine isn't working
300     - what year is it
301     - order a pizza
302     - I want to order a pizza
303     - I want to know the weather
304     - what is the weather today
305     - what is the weather
306     - what the weather today ?
307     - hows the weather
308     - tell me a joke
309     - Can I get a hamburger?
310     - Can YouTube talk?
311     - Can you call me back ?
312     - Can you give me your datacenter's password
313     - Can you give me your datacenter's password?
314     - Can you make sandwiches?
315     - Can you please send me an uber
316     - ofif fsdfafsfs
317 - intent: hold_story
318   examples: |
319     - wait
320     - stay there
321     - hang on
322     - hang on their
323     - give me a minute
```

Figure 6 : NLU dataset “out-of-scope”

Purpose of this chatbot development is to handle account related queries in banking domain. But users may attempt to feed some irrelevant queries that related to distinctive domains like mathematical calculation, weather reporting, and online requesting frameworks. As an intelligent banking assistant, when user request any out of scope queries chatbot still need to address those queries in appropriate manner. Users have some general idea about early chatbots that develop related to weather reporting, online ordering systems and entertaining like storytelling. Therefore, if they feed any out-of-scope queries chatbot need to determine the query using trained NLU dataset.

Figure 6 shows the out-of-scope NLU dataset. When the model trains the out-of-scope dataset it will help exhibit chatbots intelligence by providing convenient answers without fall back the situation. Maintaining conversation flow between user and the banking assistant is a primary concern. To smooth out the conversation flow, trained model needs to determine the out of scope and other irrelevant queries separately. Handling those queries without interrupting into the main conversation is pivotal.

```
! nlu.yml x
data > ! nlu.yml > [ ] nlu > { } 23 > synonym
345 - synonym: fixed
346   examples: |
347     - fixed deposit
348     - FIXED ACCOUNT
349     - FIXED
350     - fixed account
351     - fixed accounts
352     - fixeed
353     - Fixed
354     - Fixed account
355 - synonym: current
356   examples: |
357     - Current deposits
358     - CURRENT
359     - CURRENT ACCOUNT
360     - current account
361     - current accounts
362     - Current
363     - Current Account
364     - current
365     - Current account
366 - synonym: Kekulu Smart
367   examples: |
368     - Kekulu SMART
369     - KEKULU SMART
370     - Kekulu Smart
371     - Kekulu
372     - kekulu Smart
373     - kekulu smart
374     - kekoolu smart
375     - kekoolu
376     - kekuloo smart
377     - Kekuloo Smart
```

Figure 7 : NLU dataset “synonym”

Different users may represent the same context by using different words. For example, fixed, fixed deposit, fixed account, and fixed accounts all representing the context of fixed accounts. Therefore, the chatbot model required training data with synonyms. No matter what word that user fed, chatbot needs to recognize the synonym and use the labeled word for further operations. Once the entity extractor extracts the entities for a sentence and if it contains a synonym extractor will consider the labeled word.

This will be effective when querying the database. When a user requests details about Fixed accounts, chatbot uses the synonym label as the key word for querying the database. If the chatbot model utilizes a user given misspelled word as key word and runs the database query, the result will certainly be a false answer. Figure 7 the some Synonym words that are used in the NLU dataset.

3.3.3 Training Stories Design

Stories are form of training data used to train the dialogue management models. A story is a representation of an actual conversation between a user and an ai assistant, converted into a specific format where user inputs are expressed as corresponding intents and entities where

necessary while the responses of an assistant are expressed as corresponding action names (“How to Design Rasa Training Stories | Rasa Blog,” 2019).

```
! stories.yml X
data > ! stories.yml > [ ] stories > { } 1 > [ ] steps
3  stories:
4
5  - story: search product types
6    steps:
7      - intent: greet
8      - action: utter_how_can_i_help
9      - intent: search_account_types
10     entities:
11       - account_type: savings
12     - action: action_account_type_search
13     - slot_was_set:
14       | - products: "Savings"
15     - intent: thanks
16     - action: utter_goodbye
17
18  - story: search product + account types
19    steps:
20      - intent: greet
21      - action: utter_how_can_i_help
22      - intent: search_account_types
23      - action: utter_ask_account_type
24      - intent: inform
25     entities:
26       - account_type: savings
27     - action: action_account_type_search
28     - slot_was_set:
29       | - products: "Savings"
30     - intent: thanks
31     - action: utter_goodbye
```

Figure 8 : Training stories “search products

As we discussed in the previous section, the same query can be asked in different ways. Figure 8 shows how to handle those situations. If a customer inquire “what are the Savings accounts available in the bank”, this query includes intent and the entity that customer is looking for. Intent is searching for products and entity is savings accounts. In this scenario chatbot should response with the available savings accounts in the bank. No need to verify the account type that the customer needs to explore. This scenario shows in the **search product types** story.

In addition to that, customer may inquire about the same query without providing a specific account type. To handle this situation chatbot needs to train with the stories like **search product + account types**. When the customer sends the query without an entity, “what are the account available in the bank?”. The chatbot needs to classify the intent and needs to request the account type that the customer is attempting to explore.

Most common scenario of the conversation is middle of the conversation the chatbot is required to collect a few pieces of information from the user. To collect information from the user we have to use Rasa forms as actions. Figure 9 shows that form action used in Suggesting account

story. In order to suggest an account to the user, chatbot needs to collect few information from the user.

First chatbot request account type that user interest and after filling the account_type slot, chatbot next request the age group from the user to fill age_group slot. After filling those two slots chatbot querying database to find convenient accounts that related to given information.

```
! stories.yml x
data > ! stories.yml > [ ] stories > { } 2 > [ ] steps
71 - story: suggesting account
72   steps:
73     - intent: greet
74     - action: utter_how_can_i_help
75     - intent: search_products
76     - action: select_account_form
77     - active_loop: select_account_form
78     - slot_was_set:
79       | - account_type: "savings"
80     - slot_was_set:
81       | - age_group: "youth"
82     - active_loop: null
83     - action: action_select_account_related_to_age
84     - intent: thanks
85     - action: utter_goodbye
86
87 - story: search product + account types + serch account
88   steps:
89     - intent: greet
90     - action: utter_how_can_i_help
91     - intent: search_account_types
92     entities:
93       | - search_account: tell me
94     - action: utter_ask_account_type
95     - intent: inform
96     entities:
97       | - account_type: savings
98     - action: action_account_type_search
99     - slot_was_set:
100    | - products: "savings"
101    - intent: thanks
102    - action: utter_goodbye
```

Figure 9 : Training stories “suggesting account”

3.3.4 Domain Design

The domain defines the universe or the field in which the chatbot operates. (“Domain,” n.d.) Domain file is the master file of the chatbot that declares the intents, entities, forms, slots, actions, and responses that chatbot would know about. All the intents, entities and slots that contain in the NLU dataset needs to be declared in the domain file.

Addition to that all the responses have to be included in this file and one or more responses can have one category (utter). If an utter includes more than one response, chatbot will use the responses randomly. It will improve the quality of conversation by responding with diverse

answers with similar meaning. Next important component is custom actions. Domain file needs to consist of custom actions that chatbot need to be aware with.

Figure 10 shows the responses that include the domain file. “Utter_how_can_i_help” is one of the responses in the domain file. We can include multiple responses to domain file as we discussed in earlier. After training this dataset, the chatbot is able to select responses randomly and use different responses to the different situations. As an example, every time that the user says “Hi”, chatbot needs to respond with different approaches otherwise the user will feel informal. Therefore, chatbot model training with multiple responses will contribute an added advantage for convenient conversation between user and the chatbot.

```
! domain.yml ×
! domain.yml > {} responses
60 responses:
61 utter_goodbye:
62 - text: Bye
63 - text: Come again..
64 - text: Good Bye
65 utter_iamabot:
66 - text: I am a bot, powered by Rasa.
67 - text: |
68 | I am a Banking Bot,
69 | And I can help with any account related queries.
70 utter_how_can_i_help:
71 - text: Hello, I am a Virtual Banking Assistant. I can help you with any Account related queries
72 - text: Hi, My name is Banking Bot. How can I help you today?
73 - text: Hello, I am Banking Bot. I can help you with any Account related queries
74 utter_ask_account_type:
75 - text: Can you provide the account type that you want to explore. We have Current, Savings, and Fixed deposit Accounts
76 - text: |
77 | We have Current, Savings, and Fixed deposit Accounts.
78 | Which one you need to explore?
79 utter_ask_age_group:
80 - text: Please share us the Age group you are looking. Minor, Youth, General,or Senior Citizen
81 utter_account_closing_procedure:
82 - text: |
83 | If you want to close your {account_type} account. Please visit your nearest branch and contact Personal Banking Department.
84 | NOTE: NIC or Driving license is Mandatory
85 utter_product_type:
86 - text: Which Product type are you referring? eg. Kekulu Smart, TSB Prestige, Smart FD, etc.
87 utter_which_account_type:
88 - text: Savings account Interest Rates or Fixed Deposits Interest Rates?
89 utter_account_openning_procedure:
90 - text: |
91 | If you want to open an {account_type} account. Please visit your nearest branch and contact Personal Banking Department.
92 | NOTE: NIC or Driving license is Mandatory
```

Figure 10 : Domain “responses”

As I mentioned in previously, domain file contains intents, actions, and forms. Forms are important when responding to the customers by using their own data collected from the conversation. In the figure 11 shows a form that is used to collect account type and the age group from the user and respond to the user using his own data.

```
! domain.yml X
! domain.yml > [ ] actions
6  forms:
7  select_account_form:
8  required_slots:
9    account_type:
10   - entity: account_type
11     type: from_entity
12   age_group:
13   - entity: age_group
14     type: from_entity
15
16  actions:
17  - action_account_type_search
18  - action_default_ask_affirmation
19  - action_initial_deposit
20  - action_interest_rate
21  - action_interest_rate_product
22  - action_select_account_related_to_age
23  - action_service_charge_inquiry
24  - action_show_products
25  - utter_account_closing_procedure
26  - utter_account_opening_procedure
27  - utter_ask_account_type
28  - utter_goodbye
29  - utter_how_can_i_help
30  - utter_product_type
31
32  intents:
33  - search_products
34  - greet
35  - inform
36  - goodbye
37  - banks_products
38  - initial_deposit_amount
```

Figure 11 : Domain “forms”

3.3.5 Custom Actions

When the bank assistant predicts a custom action, the Rasa server sends a POST request to the action server with a json payload including the name of the predicted action, the conversation ID, the contents of the tracker and the contents of the domain. After completing the custom action, it returns a json payload of responses and events. (“Actions,” n.d.)

The figure12 shows a custom action that banking bot use to search account types that are available in the bank. This action connects to the database and collects products relevant to the extracted account type (savings, fixed or current) and retrieves those data as button format.


```
actions.py ×
actions > actions.py
22 # class Search account Types in the DB
23 class ActionAccountTypeSearch(Action):
24
25     def name(self) -> Text:
26         return "action_account_type_search"
27
28     def run(self, dispatcher: CollectingDispatcher,
29             tracker: Tracker,
30             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
31
32         facility = tracker.get_slot("account_type")
33         ob = databaseConnection()
34         products =ob.searchProduct(facility)
35
36         if bool(products) == False:
37             dispatcher.utter_message(text="Sorry, please provide the account type that you want to explore." )
38         else:
39             buttons=[]
40             for c in products:
41                 name = functools.reduce(operator.add,(c))
42                 print(type(name))
43                 payload = "/banks_products{\"product\": \"\" + name + "\"} "
44                 buttons.append({"payload":payload,"title":c})
45
46             dispatcher.utter_message(text="Here is Our product in related to account :", buttons=buttons)
47
48     return()
```

Figure 12 : Custom action “account type search”

When a user request a certain account type like fixed, savings or current, the chatbot needs to run “action account type search” custom action. Custom action will receive relevant account type and using that information action will query the bank database and anticipate relevant product types from the database. When the action receives the product types, it will present to the customer as an array of buttons. The user will receive his\her expected response and in addition to that they can click on the buttons that they wish, and they can find more detail about the bank product. Figure 13 shows the database connection class in the action module.

```

actions.py X
actions > actions.py > databaseConnection > selectProduct
301 class databaseConnection:
302
303     def __init__(self):
304         self.mydb =sqlite3.connect('bankdb.db')
305
306     def searchProduct(self,account_type):
307         try:
308             mycursor = self.mydb.cursor()
309             mycursor.execute("select name from products where account_type='%s'"% account_type)
310             print(account_type)
311
312             myresult = mycursor.fetchall()
313         except sqlite3.Error as error:
314             return("An exception occurred in Collecting data")
315         else:
316             for row in myresult:
317                 print(row)
318
319             return(myresult)
320
321     def selectProduct(self,account_type,age_group):
322         try:
323             mycursor = self.mydb.cursor()
324             mycursor.execute("select name from products where account_type='%s' and age_group='%s'"% (account_type,age_group))
325             print(account_type)
326
327             myresult = mycursor.fetchall()
328         except sqlite3.Error as error:
329             return("An exception occurred in Collecting data 1")
330         else:
331             for row in myresult:
332                 print(row)
333             return(myresult)

```

Figure 13 : Custom action “database connection”

```

actions.py X
actions > actions.py > ActionDefaultAskAffirmation > run
259 class ActionDefaultAskAffirmation(Action):
260     """Asks for an affirmation of the intent if NLU threshold is not met."""
261
262     def name(self):
263         return "action_default_ask_affirmation"
264
265     def __init__(self):
266         self.intent_mappings = {}
267         # read the mapping of 'intent and valid question' from a csv and store it in a dictionary
268         with open(
269             INTENT_DESCRIPTION_MAPPING_PATH, newline="", encoding="utf-8"
270         ) as file:
271             csv_reader = csv.reader(file)
272             for row in csv_reader:
273                 if row:
274                     self.intent_mappings[row[0]] = row[1]
275
276     def run(self, dispatcher, tracker, domain):
277         predicted_intent_info = tracker.latest_message["intent_ranking"][1]
278         # get the most likely intent name
279         intent_name = predicted_intent_info["name"]
280         # get the prompt for the intent
281         intent_prompt = self.intent_mappings[intent_name]
282
283         message = "Did you mean '{}'?".format(intent_prompt)
284
285         buttons = [
286             {"title": "Yes", "payload": "/" + intent_name},
287             {"title": "No", "payload": "/out_of_scope"},
288         ]
289
290         dispatcher.utter_message(message, buttons=buttons)

```

Figure 14 : Custom action “intent description mapping”

Custom actions can be used to customize existing default actions. We have used two stages fall back policy to the banking chatbot to improve the quality of conversation. Therefore, need to map intents into user-friendly terms. Description mapping.csv file contains intents and convenient user-friendly user message related to intents. This custom action activates when the chatbot predicts intent with low probability value. When the chatbot predicts such a low probability value, two stage fall back policy will provide low probability intents to the user and request whether they are meant for that intent or not. But users are unable to identify the intent name. Therefore, this custom action will select the user-friendly message from the csv file and show that message to the user without showing intent name. Existing default actions can be customized using custom actions as shown in figure 14.

```
! config.yml x
! config.yml > [ ] policies
3  language: en
4
5  pipeline:
6    - name: whitespaceTokenizer
7      case_sensitive: false
8    - name: RegexFeaturizer
9    - name: LexicalSyntacticFeaturizer
10   - name: CountVectorsFeaturizer
11   - name: CountVectorsFeaturizer
12     analyzer: char_wb
13     min_ngram: 1
14     max_ngram: 4
15   - name: DIETClassifier
16     epochs: 200
17     constrain_similarities: true
18   - name: EntitySynonymMapper
19   - name: ResponseSelector
20     epochs: 200
21     constrain_similarities: true
22   - name: FallbackClassifier
23     threshold: 0.7
24     ambiguity_threshold: 0.1
25
26  policies:
27    - name: MemoizationPolicy
28    - name: TEDPolicy
29      max_history: 5
30      epochs: 200
31      constrain_similarities: true
32    - name: RulePolicy
33
```

Figure 15 : Config “pipeline and policies”

All the pipeline and policies-based configurations are mentioned in the config file as shown in figure 15. We have used white space tokenizer for the tokenization unit. It will separate the sentence into chunks (words) and move to the featurizer unit. further that we have disable the case sensitivity in the tokenzer to gain accurate predictions. When case sensitivity enabled, same word with capital letters and simple letters will determine as different entities. There are three featurizers are used in the chatbot, those are Regex featurizer, Lexical syntactic featurizer, and Count vectors featurizer. Separated words moved into count vector featurizer and it will

count the number of times that words appear in the bag of words. According to the vector count it will predict the intent of the user query.

In addition to that we have used Entity synonym mapper to map synonyms that consist of NLU dataset. Response selector will predict the response that needs to be displayed. Fall back classifier will give a chance to figure out what the user required and clarify user query. This fallback process used to handle situations like low confidence in NLU.

To enable fallback classifiers, we have to add Rule policy into policies configuration. Memoization policy will memorize the stories in the training data and verify if the user conversation matches the stories that consist of stories file. Then the Memoization policy will predict the next action according to matching stories of training data. Next action prediction and entity extraction will be enabled with TED policy (Transformer Embedding Dialogue). To predict next action, the model needs to look at dialogue history and Max history will limit the number of dialogues that model looks into.

```
! rules.yml x
data > ! rules.yml > ...
1  version: "2.0"
2
3  rules:
4
5  - rule: Say goodbye anytime the user says goodbye
6    steps:
7      - intent: goodbye
8      - action: utter_goodbye
9
10 - rule: Say 'I am a bot' anytime the user challenges
11   steps:
12     - intent: bot_challenge
13     - action: utter_iamabot
14
15 - rule: out of scope
16   steps:
17     - intent: out_of_scope
18     - action: utter_out_of_scope
19
20 - rule: Implementation of the Two-Stage-Fallback
21   steps:
22     - intent: nlu_fallback
23     - action: action_two_stage_fallback
24     - active_loop: action_two_stage_fallback
```

Figure 16 : Rules

Rules are another sort of training data that used to train chatbot model and also, they characterize short pieces of conversations that model need to follow as the same. As shown in figure 16, chatbot model uses rules training data to predict next action when handling out of

scope queries, goodbyes, bot challenge queries and two stage fall backs. As an example, whenever a user says goodbye chatbot need to respond with a goodbye. In the two-stage fallback process also chatbot model needs to follow the fallback steps one by one. If the intent is recognized as NLU fallback, then the next prediction will be a fallback action loop.

3.3.6 Deployment of AI assistant

Deployment makes sure the AI assistant is accessible to users and that it is configured in a production ready environment. We have created a Virtual Machine in Google Cloud Platform to deploy both Rasa X and the AI assistant. Rasa X is a Conversation-Driven Development (CCD) platform that allows listen to user conversation and apply those to improve the AI assistant. Figure 17 shows the GCP VM instances that created to deploy the AI assistant.

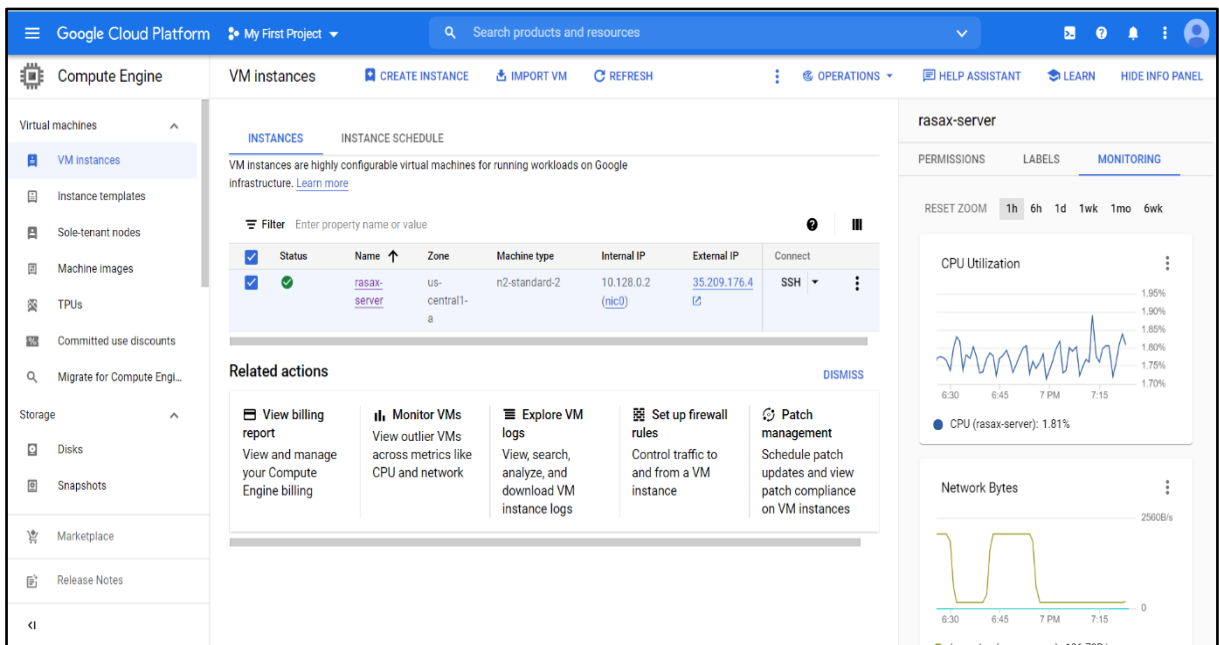


Figure 17 : VM instance in Google Cloud Platform

After completing the Rasa X deployment, the AI assistant is able to deploy and configure the settings inside the Rasa X Conversation-Driven Development (CCD) platform. Rasa X facilitates the external channel deployment for the AI assistant. We have integrated a web interface and slack messaging channels to the AI assistant using Rasa X. Addition to that, Rasa X provides Continuous integration and Continuous Deployment (CI/CD) to improve the AI assistant. Figure 18 shows the interface of the Rasa X platform.

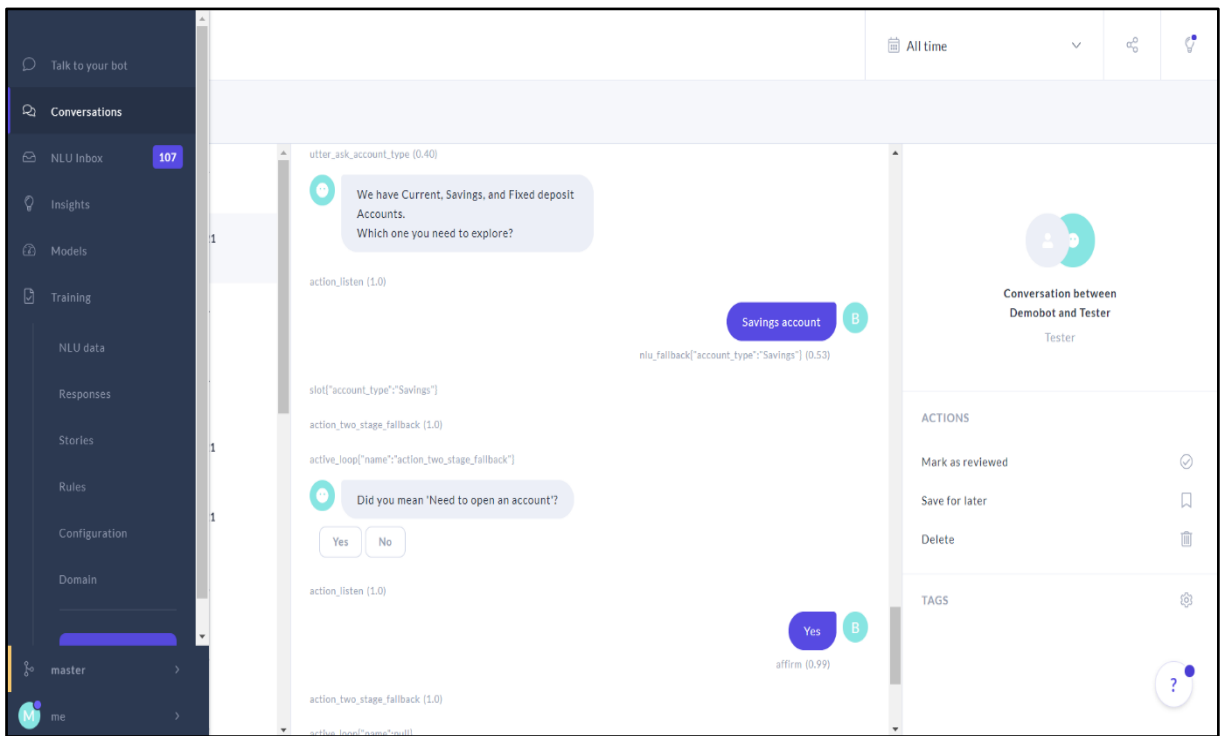


Figure 18 : Conversation History in Rasa X Platform

CHAPTER 4: RESULTS AND EVALUATION

In this section, we will cover the results of the banking assistant model and evaluation process that were used in banking assistant development. Real user evaluation is critical for these kinds of projects. Reason for that is after the implementation, the product will use real-world customers. Participant and the two evaluation cycles are the main areas that are discussed through this chapter.

4.1 Results

Final expectation of the research is to develop an advanced Artificial Intelligent banking assistant that is capable of answering account related customer queries. And the trained model is able to connect with different interfaces like slack and web interface. Having multiple channels to reach a banking assistant is convenient to draw customer attention with different social states. Most importantly, the final product of AI banking assistant is capable of handling account related user queries, out of scope queries and when necessary, situations occur, the AI assistant is trained to fallback gracefully.

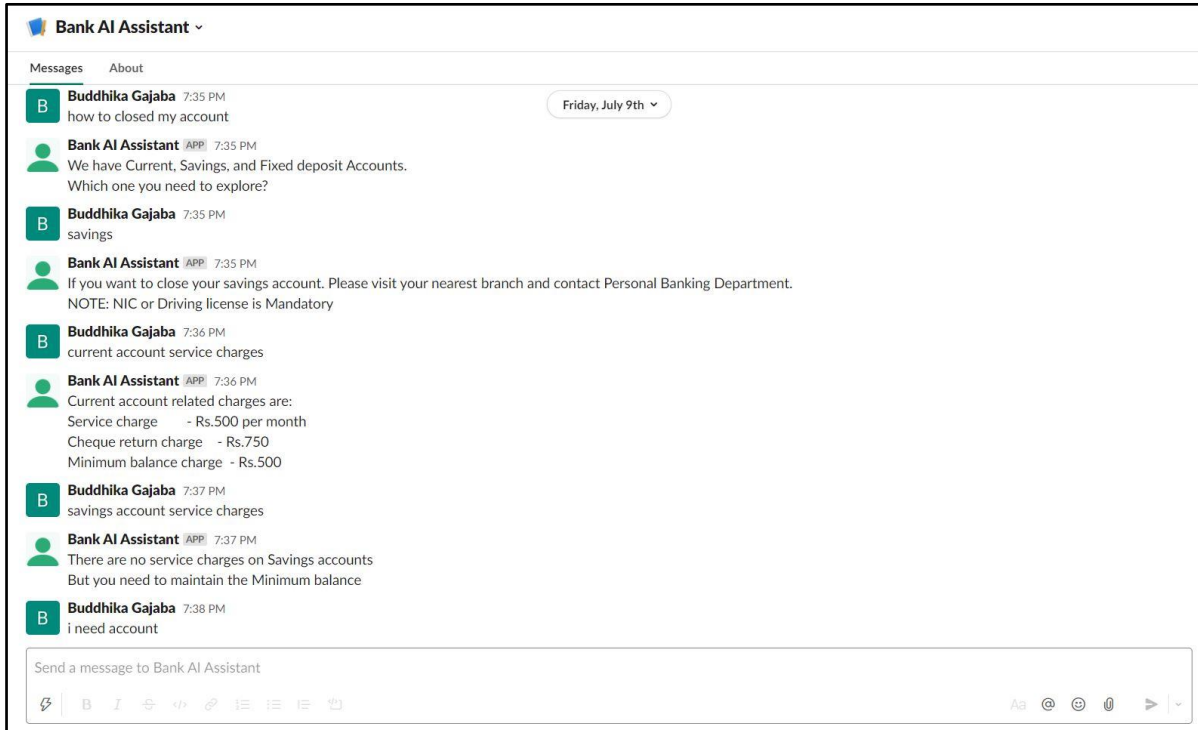


Figure 19 : Interface “slack messaging platform”

Banking AI assistant have integrated with slack framework as well as web interface. In the figure 19 shows the Banking chatbots responses in the slack interface. Figure 20 shows a screenshot of the web interface. Chatbot model able to respond to both these interfaces independently. Therefore, users who are familiar with slack, able to reach Banking assistant via the slack messaging platform and other users can use the bank web interface.

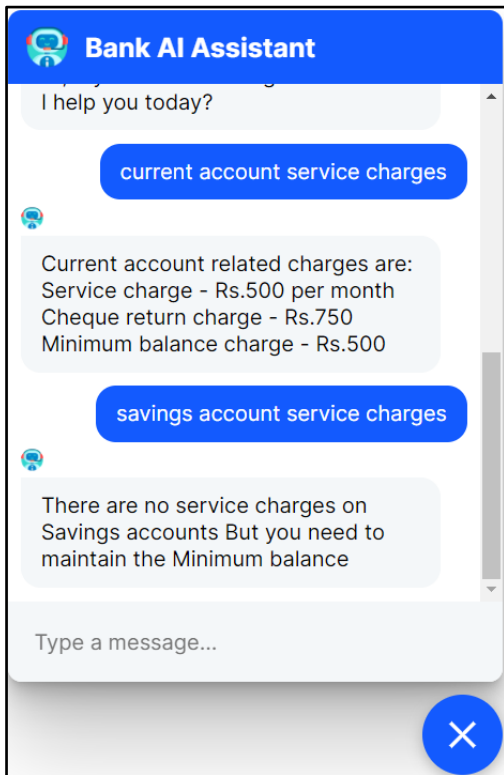


Figure 20 : Interface "web interface"

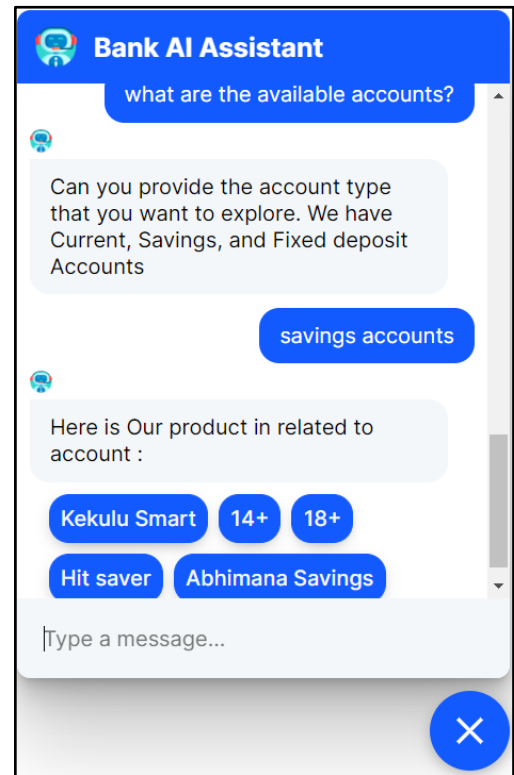


Figure 21 : Respond "account search"

As I mentioned in the implementation chapter, when a user requests details about available accounts, the AI assistant will require additional information from the user. To acquire the information, the banking assistant is requesting the account type that the user attempts to search. After receiving the account type, the AI assistant will generate products that are available in the bank by querying the database. Figure 21 shows the mentioned scenario.

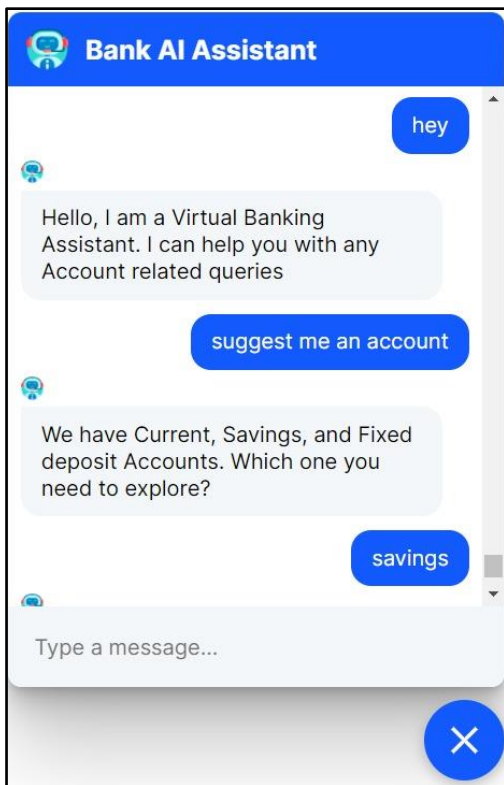


Figure 22 : Respond "suggesting account I"

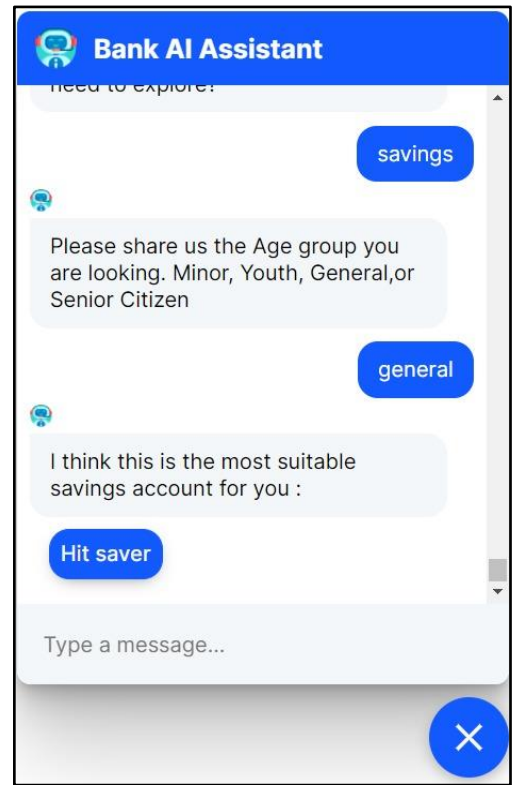


Figure 23 : Respond "suggesting account II"

Figure 22 and 23 shows the account suggestion feature that banking assistant is capable of. In this scenario AI assistant model use form action to collect users desires and using that information model will query the database and show a convenient product for the user. AI assistant will collect the user desired account type and the age group for this process. Chatbot model has trained with NLU training data that contain "inform" intent to recognize those two entities accurately.

Figure 24 shows the two-stage fallback. As we discussed in the previous chapters, the crucial requirement of the AI assistant is to handle low confidence intents. It is more honored that verifying the predicted intent from the user than requesting the user to rephrase the sentence. Chatbot model unable to predict intent with high confidence rate in every time. Therefore, if the AI assistant needs to fall back the situation assistant is able to fallback gracefully.

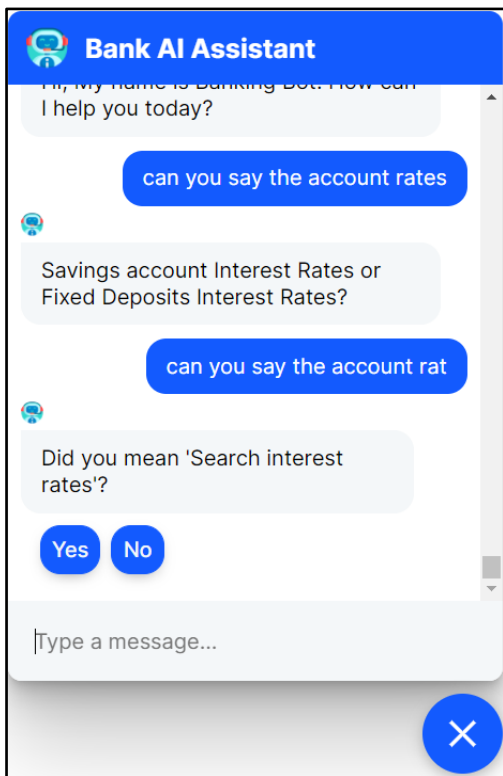


Figure 24 : Responds "2 stage fallback"

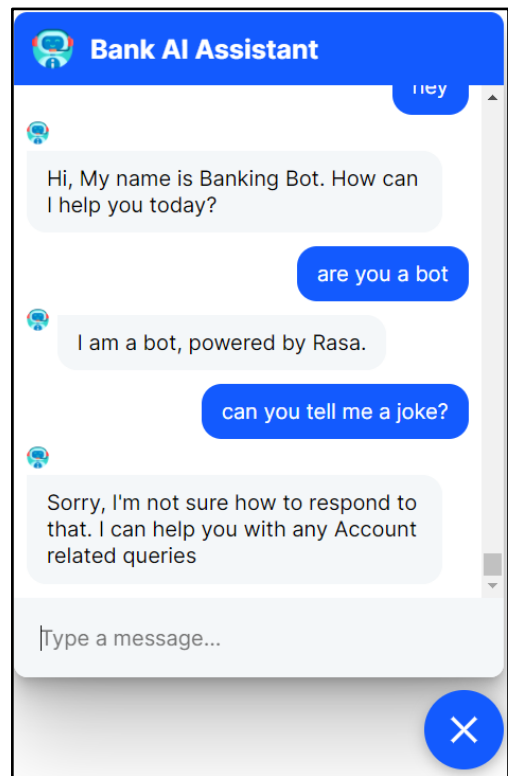


Figure 25 : Responds "out-of-scope"

Users are able to ask irrelevant queries from the AI assistant, but bot needs to respond to them related to the question that user asked. Figure 25 shows two different situations that bank assistant manages the situation smoothly. In the first query, the user is trying to verify that he communicates with a real person or AI assistant. In the model training stage, we have trained the AI model using NLU training dataset consisting of different chatbot challenge intents. In the second query, the user requests an out-of-scope requirement. Because of that, AI assistant fallback the situation and respond with its capable features.

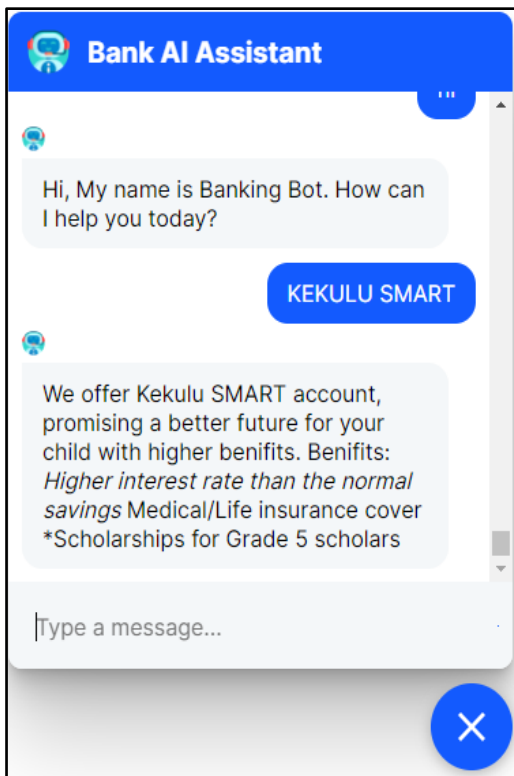


Figure 26 : Responds "synonym"

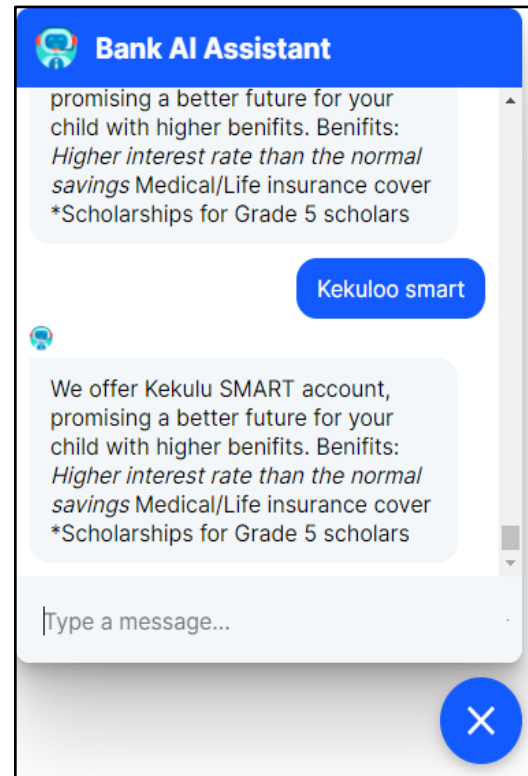


Figure 27 : Responds "synonym Misspell"

Model that training with the synonyms in NLU data help to handle misspelled words in the user query. Correctly spelled situation is shown in the figure 26 and misspell handled situation shows in the figure 27. When a user trying to reach a product type using a query, there is a possibility that the user will misspell the product name. Therefore, model have to train with various different possible words in order to handle this kind of situation.

4.2 Evaluation

4.2.1 Evaluation approach

Primary objective of the research is to implement an Artificial Intelligence banking assistant to answer account related customer queries. We must evaluate the chatbot's responses in order to reach this target. In the chatbot context we have to evaluate several criteria.

- Chatbot responses are relevant to the user's query
- Capability of maintain conversation flow smoothly
- Responds to outer scope queries
- Handling fallbacks gracefully

In order to evaluate the chatbot, we have to collect real user experience from the users. The feedback should be collected from the users who have done a sufficient involvement with the chatbot. Users are able to share their experience by completing google form that contains a questionnaire (Annexure 1).

This questionnaire includes matrix and rating type questions and open-ended questions. These matrix and rating type questions are capable of presenting the rating scale in a logical or consistent order. Therefore, it makes sense to order the ranking or rating choices from low to high (Very poor, Poor, Average, Good, Excellent). In the first stage of implementation these matrix and rating questions are very useful to evaluate the present status of the Banking chatbot and identify the areas that need to improve.

We have evaluated the responses related to different account types, how to use greetings and how to respond to out of scope. Questionnaires consist of matrix and rating type questions to evaluate those responses. In addition to those open-ended questions provide present failures and new suggestions that real users are expecting. Therefore, open-ended questions provide ideas to expand chatbot further.

4.2.2 Participant

The Banking chatbot evaluation is done by banking and non-banking users. The banking users will know their products and procedures followed by the bank itself and other common users do not have any idea about bank products or the bank procedures. To ensure both the users can have the same experience from the banking chatbot is the main goal of this project and that is why we have used bank and the non-bank users for the evaluation process. There are 10 banking users, and 15 non-banking users are the participants in this evaluation process. From the beginning of the implementation, we have motivated them to use the prototype and provide their feedback using google form. In this chapter, the results of the evaluation are presented and discussed.

4.2.3 Evaluation on Prototype 01

After the first prototype implementation, users could use the chatbot via slack and the web interface. After having reasonable conversation with the Banking chatbot users have provided their opinion and feedback using provided google form (Annexure 1).

4.2.3.1 Prototype 01 Evaluation results

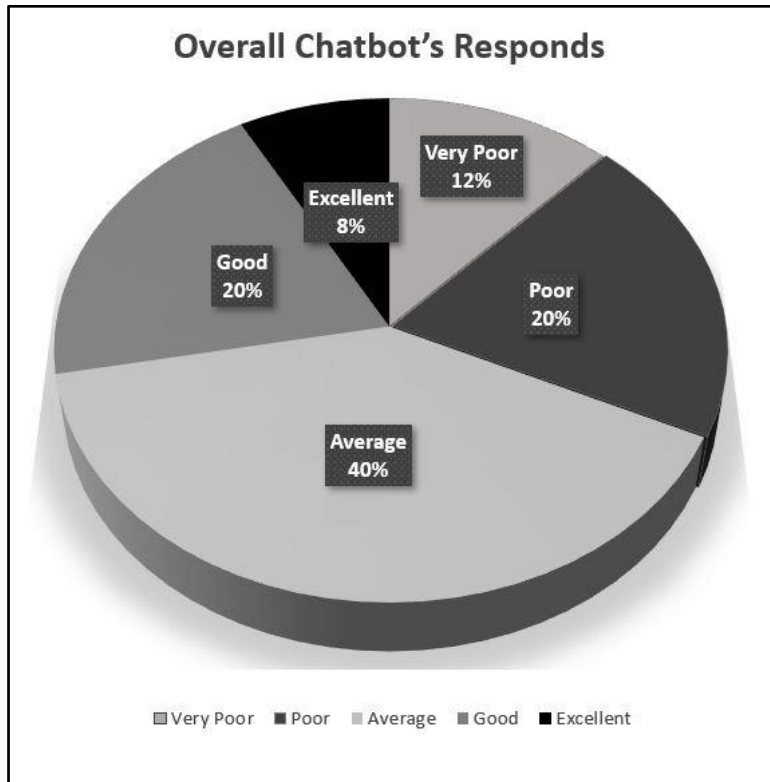


Figure 28 : Overall chatbot evaluation – prototype 01

Participants gave this result after having conversation with a banking chatbot and considering overall responses that chatbot provided. Figure 28 shows the overall chatbot evaluation results. According to figure 10, 12% of participants rank chatbot with a very poor rating, 20% gave poor, 40% gave Average, 20% gave Good, and 8% gave Excellent rating. That means 72% of participants gave an average or poor rating on the first prototype. To identify the reason for this result we have to use evaluation results with responses for each category (figure 29).

According to the results shown in figure 29, responses for the greetings and savings account related queries are in respectable state but rest of the responses need to improve to achieve the expected standard. Responds that generated related to Current account, fixed account related and out of scope were misguided. Reasons for that misguidance were missed classification of each intent. In order to rectify the issue, we have to reinforce the NLU dataset with new divergent data.

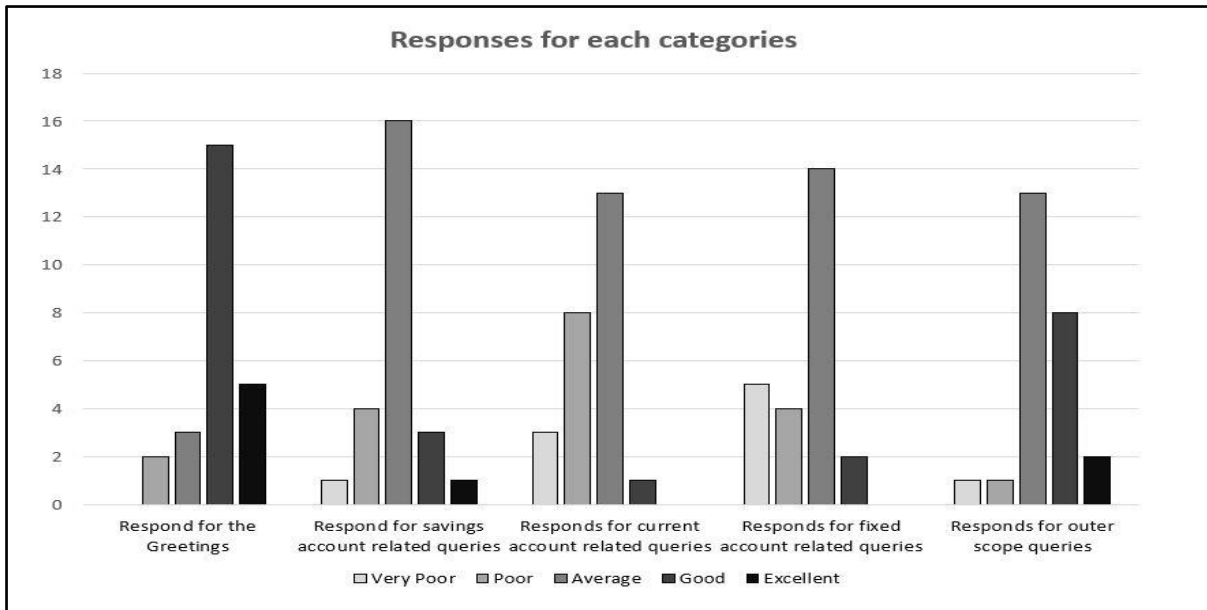


Figure 29 : Evaluation results category wise - prototype 01

To improve the accuracy level of intent classification, we had to increase the NLU data set accordingly and needed to specify entities accurately. Customer queries included with some capital letters did not identify correctly. For example, “I need to know about Fixed accounts”, in this scenario a user enquiring about fixed deposit accounts but chatbot does not classify Fixed accounts because of when capital letters are included with the sentences. In addition to that, participants have used similar words to inquire about products. When chatbot classify intents incorrectly chatbot will provide invalid response. This fault becomes the reason to have average results on the first evaluation. To avoid missed classification we have used synonyms words in the NLU and trained the chatbot model in the second development cycle.

In addition to rating participants gave their new ideas and mentioned areas that need to improve in the banking chatbot.

- Provide account (product) benefits in addition to the account details.
- Add information about Credit cards.
- Provide ATM locations nearby.
- Internet banking related details
- ATM or CRM related problem handling
- Current account related details. e.g., Overdrafts, check books.
- Fixed deposit renewing procedure.

4.2.4 Evaluation on Prototype 02

As we discussed in the previous section, we had to develop an NLU data set to accurate the intent classification and use synonyms to increase accuracy on entity extraction. For example, participants used different words to represent “savings account” like “savings”, “saving”, “savings account”, “saving accounts”. More importantly participants are provided extra comments providing new ideas and they mentioned areas that need to improve. We have selected a number of areas and enhanced the banking chatbot. Some recommended areas like credit card detail and areas like ATM locations are out of my project scope. Therefore, those areas need to cover up with future work.

Selected areas to develop:

- Provide account (product) benefits in addition to the account details.
- Internet banking related details
- Current account related details. e.g., Overdrafts, check books.
- Fixed deposit renewing procedure.

After the second development iteration is completed, we have done the same evaluation procedure again. Used the same 25 participants and provided an upgraded chatbot prototype. After having sufficient involvement with chatbot prototype users have to answer the google evaluation form.

4.2.4.1 Prototype 02 Evaluation results

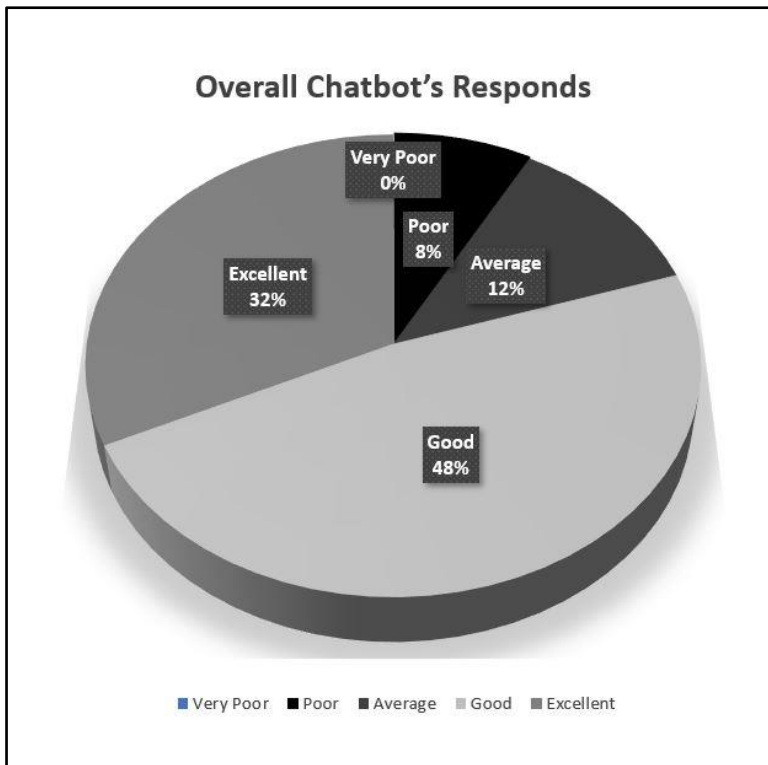


Figure 30 : Overall chatbot evaluation – prototype 02

At the second evaluation cycle participants were familiarized with the banking chatbot and we have improved NLU data and the features that users requested. Therefore, in the second evaluation results were improved compared to prototype 01 results. According to figure 12, Participants were not ranked chatbot with rating - Very poor, 8% participant rank chatbot with Poor rating, 12% gave Average rating, 48% gave good rating, and 32% gave Excellent rating. That means 80% of participants were satisfied with the banking chatbot and rated with the good and excellent ratings. To have a better look on this result we have to use evaluation results with the responses for each category (figure 30).

According to the results shown in figure 31, responses for all the categories were improved compared to evaluation cycle one. Majority of participants rated Respond for the greetings as good and excellent. In addition to that, savings related, current account related, fixed account related, and outer scope queries also made huge improvements with the new implementations and the NLU development. Reason behind this improvement is chatbot have trained with the different circumstances and implementing requested facilities to the users.

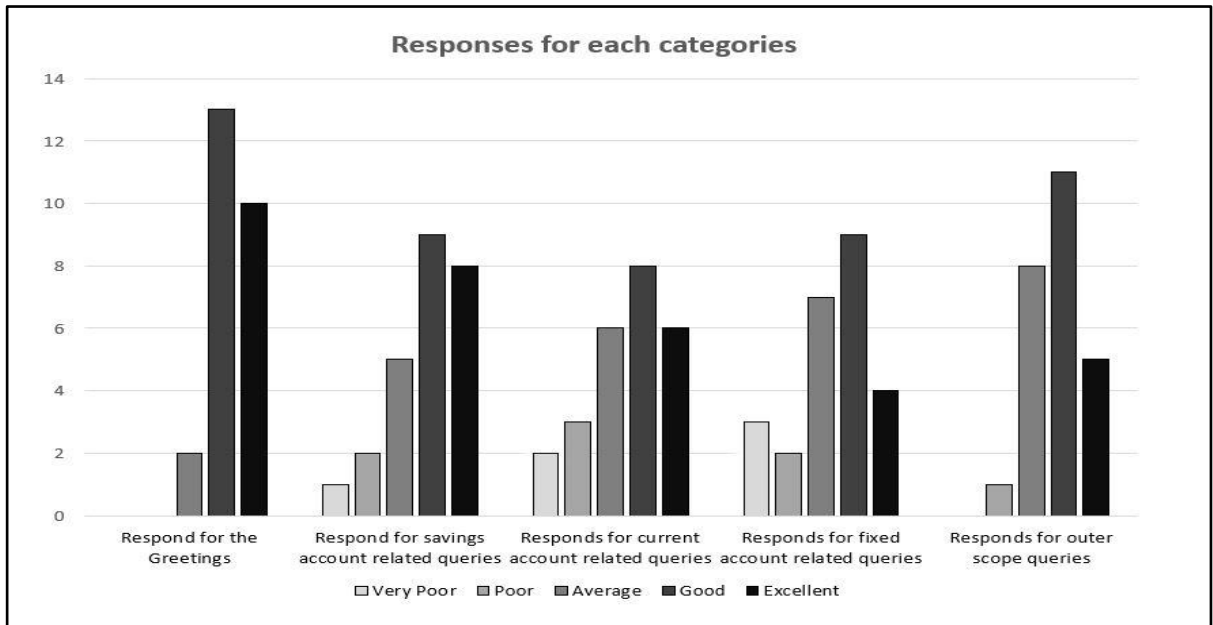


Figure 31 : Evaluation results category wise - prototype 02

4.3 NLU Testing Result

In addition to the user evaluation, we have run the NLU model testing to verify the accuracy level of the implemented AI assistant model. Intent classification is the principal process of the AI conversation model. To predict associated response to the user query, trained model needs to classify intent accurately. Therefore, testing the NLU intent classification is crucial.

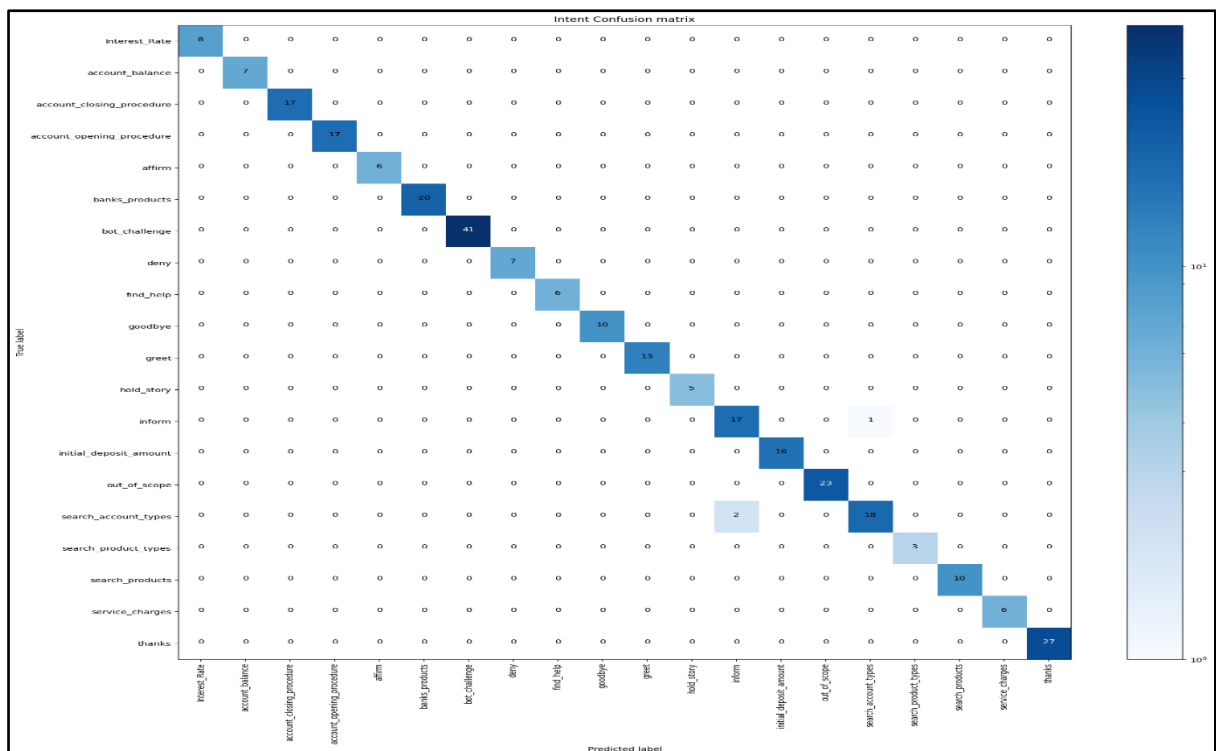


Figure 32 : Intent confusion matrix

Figure 32 shows the intent confusion matrix of the NLU dataset. This is the output of the testing process. According to the final confusion matrix all the intents are predicted correctly, and the accuracy level of the predictions are substantially high. Therefore, trained models are able to predict next action accordingly and accurately. This will ensure the accuracy of the final output of the chatbot. And figure 33 shows the intent histogram. When improving the quality of the training data will show blue histogram at the top of the plot and red histogram bars need to stay bottom of the plot. According to that explanation this histogram is in a strong position.

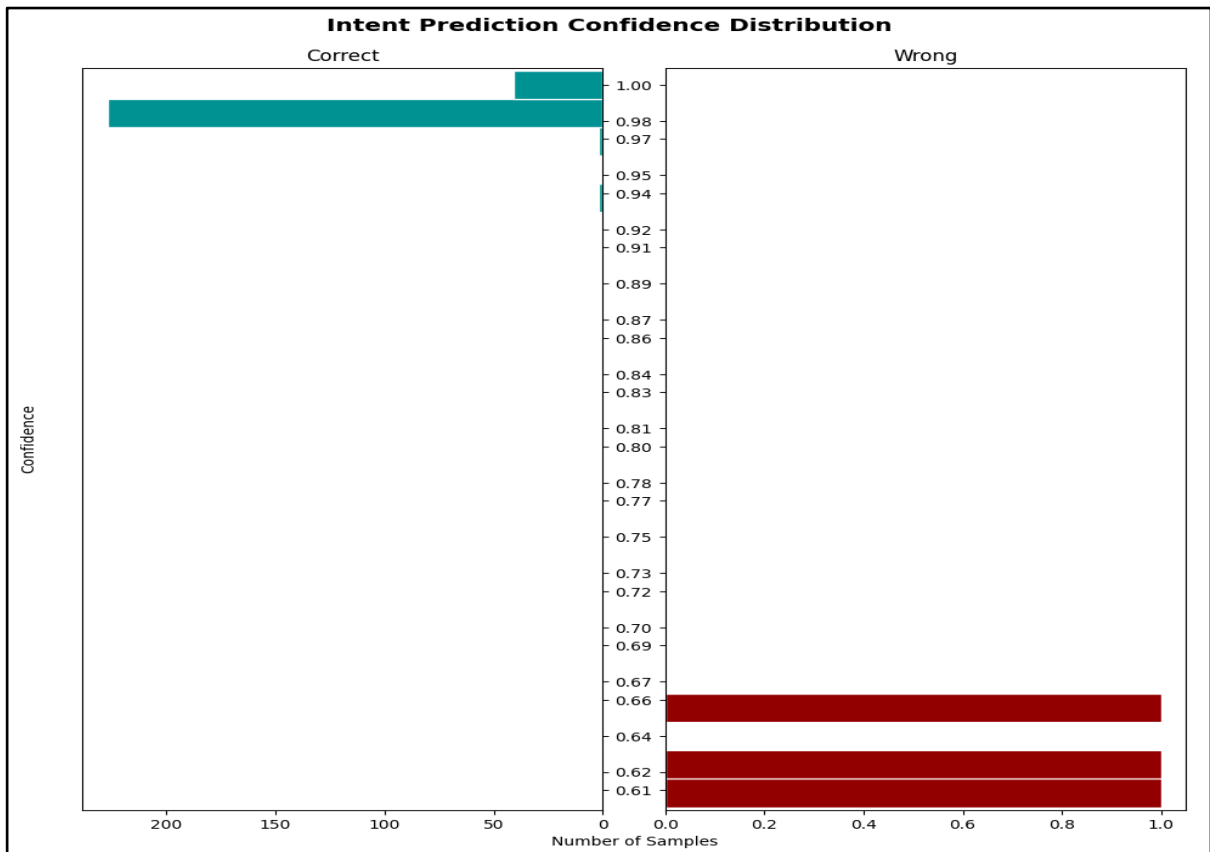


Figure 33 : Intent histogram

4.4 Summary

Evaluation process ensures the project develops on the correct path and achieves the main objective of the banking chatbot implementation. Evaluation results are more convenient way to identify positive and negative feedbacks regarding chatbot responses. After the first development iteration evaluation results provide opinion on the next development pathway. Final implementation starts considering the evaluation results on the first iteration and repeatedly evaluation process takes place after the implementation. Final evaluation results lay out the real user opinion on the banking chatbot and it helps to estimate the achievement on project objectives.

CHAPTER 5: CONCLUSION AND FUTURE WORK

This chapter will summarize the content of all previous sections in order to discuss what has been accomplished and limitations. It will give a global view of the completed system as well as directions for the future developments.

5.1 Conclusion

This project aimed to implement an AI based banking assistant to answer banking related queries and reduce customer inquiry calls to the bank staff. To achieve that objective, we have first discovered the Chatbot history, reason development of the chatbot field, and chatbot frameworks in the literature review chapter. In the literature review we have observed a research gap in the banking domain. Different types of chatbots were developed to focus on the financial area, but they had some limitations. To enhance the range of capability of chatbot we have determined to implement AI based banking assistant using Rasa framework. Rasa is an open-source framework for developing AI powered, industrial grade chatbots.

In the methodology chapter we have examined the chatbot architecture and the development methodology. To achieve the main objective of reducing customer inquiry calls, we have decided to integrate banking assistant with banking website and the slack messaging platform. These two interfaces are connected to the trained chatbot that develops using the Rasa framework. Most critical part of the chatbot development process is developing NLU training data. To understand the user queries a chatbot need to have advanced NLU training dataset. In order to develop the NLU dataset we have to familiarize the chatbot on intent classification and entity extraction. After classifying the intent correctly chatbot needs to provide a proper response to the inquiry. We have to develop stories and also needs to develop advanced utterances and actions to respond to appropriate answers. Finally, the developed chatbot model deployed on Google Cloud Platform VM and integrated chatbot with the banking website and the slack messaging platform.

After the first prototype implementation we have executed the evaluation process. We have selected total of 25 participants and 10 of them are banking users and 15 of them are non-banking users. Reason for that is, to ensure both domain experts and the ordinary users get the same experience from the chatbot. After having their first conversation with the banking chatbot 72% of participants rate the chatbot with Average rating. Participants were mentioned in the failed response they received and in addition to that provided their suggestions to enhance the

banking chatbot. Therefore, the first evaluation cycle results were productive to proceed to the second development cycle of the banking chatbot.

In order to develop understating of chatbot we have to develop NLU training dataset with a more reliable dataset in the next development process. Additionally, we have selected several suggestions that participants provided and enhance the chatbot features further. Soon after completing the second development cycle, we evaluated the chatbot using the same participants and collected the feedback from them. In the second evaluation cycle, we received a better rating for the banking chatbot, 48% of participants gave rating Good and 32% of participants were gave rating Excellent. That denote 80% of participants were satisfied with the banking chatbot responds. This final result represents that we have achieved the main objective of the project. Next developments can be achieved in the future works.

5.2 Limitation

There are certain limitations available in this project. Main limitation is the area that users can explore using chatbot. With the limited time frame, we have developed a banking chatbot that responds over any savings, fixed or current account related queries. Next limitation is unable to give personalized experience to the users. without having a bank user registration procedure chatbot unable to provide personalized responses to the users. In addition to those limitations' users have limitations with the language. In this banking chatbot project we have developed the chatbot only for the English language. Therefore, users have no choice over the language selection.

5.3 Future Works

As we discuss in the limitation section there are several future works that are essential for the AI based banking assistant in the future. In addition to that there are extra features that can be added to the current banking assistant and users can have better banking assistant solution in the future.

- Personalized responses make a tremendous difference to the conversation. Banking chatbot can train to reply with personalized responses using existing bank customer details.
- Banking AI assistant can improve advanced features with financial advice. Using existing customer's financial data, AI assistant is able to provide tips to the customer

about his/her expenses, loans, and investments.

- Banking chatbot needs to enhance its scope with new areas. Such as credit card related queries including requesting a credit limit, ATM and CRM machine related queries, and Internet Banking related queries including fund transfer issues, reset user passwords.
- Additional languages can be trained to the banking assistant in order to cater to a wider customer base. Generally, in Sri Lanka more than 75% use Sinhala language and 11% use the Tamil language. To cater their requirement, we have to train the chat bot with Sinhala and Tamil languages.

With eliminating the limitations and enhancing the current banking assistant project with new features, it can provide a deliberate body of knowledge to the conversational assistant's future.

APPENDICES

6.1 Appendices 1 – Questionnaire

Evaluation Questionnaire

Banking chatbot Performance evaluation and user experience.

Section 1 – Personal profile

1. Your Name

2. Your Gender

a. Male


b. Female

3. Your profession

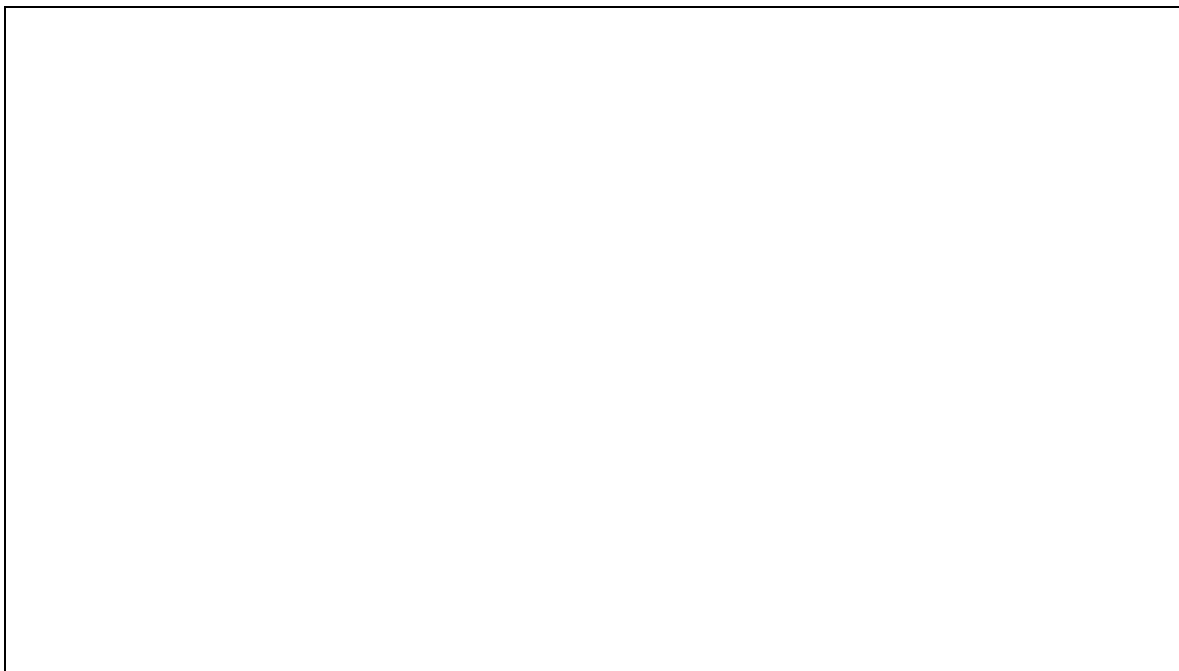
4. Please state your opinion about Banking chatbot responses according to mention areas.

	Very Poor	Poor	Average	Good	Excellent
a. Respond for the Greetings	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
b. Respond for savings account related queries	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
c. Responds for current account related queries	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
d. Responds for fixed account related queries	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
e. Responds for outer scope queries	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
f. Overall chatbot's responses	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

5. Please mention any falls responds if any.



6. Please give your suggestions to develop our chatbot further.



REFERENCES

- A Closer Look at Chatbot ALICE [WWW Document], n.d. URL <https://blog.ubisend.com/discover-chatbots/chatbot-alice> (accessed 9.2.20).
- aahill, n.d. Language Understanding (LUIS) Overview - Azure Cognitive Services [WWW Document]. URL <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis> (accessed 5.29.21).
- Actions [WWW Document], n.d. URL <https://rasa.com/docs/rasa/actions/> (accessed 8.31.21).
- AliceBot – Coming Soon, n.d. URL <https://alicebot.org/> (accessed 9.2.20).
- Chinnapa Reddy Kanakanti, Sabitha R., 2020. Ai and MI Based Google Assistant for an Organization using Google Cloud Platform and Dialogflow. *Int. J. Recent Technol. Eng.* 8, 2722–2727. <https://doi.org/10.35940/ijrte.E6354.018520>
- Designing an Emotionally Realistic Chatbot Framework to Enhance Its Believability with AIML and Information States, 2019. . *Procedia Comput. Sci.* 157, 621–628. <https://doi.org/10.1016/j.procs.2019.08.226>
- Dhyani, M., Kumar, R., 2020. An intelligent Chatbot using deep learning with Bidirectional RNN and attention model. *Mater. Today Proc.* <https://doi.org/10.1016/j.matpr.2020.05.450>
- Dialogflow Documentation [WWW Document], n.d. . Google Cloud. URL <https://cloud.google.com/dialogflow/docs> (accessed 5.29.21).
- Domain [WWW Document], n.d. URL <https://rasa.com/docs/rasa/domain/> (accessed 8.31.21).
- Dündar, E.B., Çekiç, T., Deniz, O., Arslan, S., 2018. A Hybrid Approach to Question-answering for a Banking Chatbot on Turkish: Extending Keywords with Embedding Vectors., in: *KDIR*. pp. 169–175.
- Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., Peters, M., Schmitz, M., Zettlemoyer, L., 2018. AllenNLP: A Deep Semantic Natural Language Processing Platform. *ArXiv180307640 Cs*.
- How to Design Rasa Training Stories | Rasa Blog [WWW Document], 2019. . Rasa Blog Conversational AI Platf. Powered Open Source. URL <https://blog.rasa.com/designing-rasa-training-stories/> (accessed 7.17.21).
- IBM, 2020. Watson Assistant | IBM Cloud [WWW Document]. *Watson Assist. IBM Cloud*. URL <https://www.ibm.com/cloud/watson-assistant/> (accessed 5.29.21).
- Li, B., Jiang, N., Sham, J., Shi, H., Fazal, H., 2019. Real-world Conversational AI for Hotel Bookings. *ArXiv190810001 Cs Stat*.

- Open source conversational AI [WWW Document], 2020. . Rasa. URL <https://rasa.com/>, <https://rasa.com/> (accessed 5.29.21).
- Russo-Spena, T., Mele, C., Marzullo, M., 2019. Practising Value Innovation through Artificial Intelligence: The IBM Watson Case. *J. Creat. Value* 5, 11–24. <https://doi.org/10.1177/2394964318805839>
- Samuel, I., Ogunkeye, F., Olajube, A., Awelewa, A., 2020. Development of a Voice Chatbot for Payment Using Amazon Lex Service with Eyowo as the Payment Platform. pp. 104–108. <https://doi.org/10.1109/DASA51403.2020.9317214>
- Sharma, R., 2020. An Analytical Study and Review of open source Chatbot framework, Rasa. *Int. J. Eng. Res.* V9. <https://doi.org/10.17577/IJERTV9IS060723>
- Weizenbaum, J., 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM* 9, 36–45. <https://doi.org/10.1145/365153.365168>
- Williams, J., Kamal, E., Ashour, M., Amr, H., Miller, J., Zweig, G., 2015. Fast and easy language understanding for dialog systems with Microsoft Language Understanding Intelligent Service (LUIS). pp. 159–161. <https://doi.org/10.18653/v1/W15-4622>

