A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Data Science

Chatbot for Financial Customer Support

Name: Lakshay Anand

Student ID: 001188147

Supervisor: Dr. Konstantin Kapinchev
Submission Date: 16 September 2022
Word count: 12596

Abstract

Customer service is an integral part of any running business. It basically provides customer an opportunity to be outspoken about the product and services being utilised or related desires. In modern times the quality of relationship between the customer and the company tends to be a turning point in organisation's success. Due to this reason coping up the customer's queries have become the supreme consideration. With advanced technologies that combine latest machine learning and artificial intelligence, customer management has become easy, reliable and provides an effective solution compared to traditional ways.

A proposed solution which can address and avoid the inevitable difficulties with customers by human engagement. An AI assistant mimics the conversation between the customer and the support personnel which lets organisations to deliver a cutting-edge customer experience. The assistant provides capabilities for organisations to offer continuous support, consistent answering, reduced waiting time and instant communication.

The assistant will be responsible for handling all the banking queries of the customers. The responses are delivered understanding the requirements of the user. For evaluation, I have used confusion matrix, precision, recall, and F1-score evaluation metrics.

Keywords: Chatbot, Natural Language Understanding, Artificial Intelligence, Natural Language Processing.

Acknowledgements

I would especially like to thank Dr. Konstantin Kapinchev for agreeing to be my supervisor and for his consistent advice, feedback, guidance and support throughout the lifecycle of this MSc. Data Science project.

I want to thank both Dr. Konstantin Kapinchev and Mohammad Al-Antary for agreeing to have the project demonstration on the schedule day.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

In this connected world, where every individual is utilising the internet in multiple ways as a part of their day-to-day activities. People are consuming huge amount of information that requires the retrieval of information as the similar pace. Several practices are being undertaken by institutions and agencies for helping people with information they would be searching for. Unfortunately, this assistance offered by these institutions and agencies combines human interactions for providing support to their targeted customers, through live chats and phone services. The compulsory human requirement builds a major disadvantage when it comes to assisting the clients on the massive scale; resulting is low customer satisfaction (Fauzia, Hadiprakoso and Girinoto, 2021).

Number of online applications has been developed for querying the incoming complaints of the customers. For processing a query, it is required for a customer to reach out to higher authorities or departmental admins for future development in the case conditionally the authorities are free to serve. Moreover, the process is repeated until the customer is successful in his/ her attempt. No alternative way is provided for the assistance of the customer which eventually ends up wasting precious time of the customer. Additionally, the complaint lodger directs the customers to different management staff on not being successful (Singh. et. al., 2018).

Advancement in technology has played a major part for running industry in the most efficient and productive way, these technologies not only provide the organisations the flexibility to compete with their competitions by having access over analytics and customer behaviour but additionally providing firm control within organisation. IBM created a system called Watson which is a question answering system combining technologies like advance natural language processing, information retrieval, knowledge representation, automated reasoning and lasty machine learning contributing to the open-source domain (Singh. et. al., 2018).

For coping up with these issues there are various developments to effectively built frameworks for the interaction with customers. Nowadays, Artificial Intelligence chatbots are catching some speed, which tend to be successful in eliminating the need to human interaction. A chatbot is an intelligent system that effectively communicates with the user without requiring any inclusion of any real human (Fauzia, Hadiprakoso and Girinoto, 2021).

Artificial intelligence software in chatbot gives them the capability to provide meaningful responses based on the user inputs.

Deployment of the chatbots has been seen in various industries namely: finance, retail, public and manufacturing industries that combines technologies such as Natural language processing and speech recognition which works as a personal assistant to customers communicating though an interactive interface mostly on the mobile devices. This technology has been used by several leaders of IT industry. Some of the most common examples are Apple's Siri, Amazon's Alexa, and Google's OKgoogle (Hwang and Kim, 2021).

1.1 Motivation

To addresses the issues mentioned, a smart solution indented which is capable of up the overall service probability and boomed the number of customers is the chatbot, which can efficiency handle the raised queries by the customers. Additionally, providing flexibility to drop a query at moment of time and consistent response each time on user request.

1.2 Problem Statement

Every country's economic development benefit from the presence of banks thus making them a necessity in everyone's life. Complex procedures and processes involved in the banking system makes difficult for the new customers for getting used to the system therefore not letting them to utilize the products and services to their benefit. All information related to banking and services are available on the multiple sources like mobile banking, internet banking, etc but accessing this information requires user to be technology driven. Although assistance to users is provided by the customer support team but unfortunately solving simple queries takes a massive amount of time to address them because of the long wait times and directing customers when needed. Because of this the customers tend to collect at the premisses resulting overcrowding and heated situations. Another issue faced by the banks is the repetition of the queries which involves general inquiries related to bank policies, loans, fixed deposits resulting in the non-utilisation of the resources and the manpower (Kulkarni et.al, 2017).

1.3 Research Aims and Objectives

To implement and evaluate our customer support banking chatbot which can process banking queries without requiring human intervention.

1.3.1 Aim

Aim of this project to develop a banking assistant which could handle the incoming customer queries without the indulgence of the human, resulting increased productivity to the banks. Additionally, for the customer to provide a solution in no time.

1.3.2 Objectives

Through our AI banking customer support chatbot we will achieve the following objectives:

- **Constant support** – As we are living in the digital world, everyone has access to the technology. We aim to provide customer with a consistent support by integrating our chatbot on WhatsApp.
- **Efficient banking personnel** – With the indulgence of our chatbot, the working staff at the banks could focus on the things that would be beneficial for the banks in the long period of time.
- **Mode of communication** – In this modern world, where people are seeking a faster way to do things. Our chatbot let them handles the banking queries without them being physically present at the premises.
- **Consistent answers** – Improving customer experience with consistent answers.

1.4 Scope

The scope of this research is to build an AI chatbot which can handle the customer banking queries in the most productive and consistent manner. The chatbot will be accessed via the WhatsApp which adds up the competitive advantage of availability. The chatbot will be able to handle customer queries which relates banking, borrowing, investing, and insurance. This chatbot simulates human conversation which enhances the customer experience and is purely rule based conversational agent, for maintaining consistency and efficiency in response delivery.

1.5 Structure of Thesis

This report includes the progress of the study and the work towards the final MSc. This report consists of seven chapters. Chapter 2 discusses the background of conversational agents, various models, a framework, and a few useful resources. Chapter 3 describes the design and development of a banking chatbot. Chapter 4 describes the evaluation of banking chatbot. Chapter 5 displays results of our banking chatbot in real-world setting. After that, Chapter 6 gives the conclusion and future works and lastly Chapter 7 is references.

Chapter 2: Background and Related Work

This chapter will provide essential background and architecture of a chatbot. We will also focus on the core concepts and the framework used in the development.

2.1 Conversational Agents

Conversational agents communicate with the information seeking individual by utilising natural language processing technologies, these agents are a part of the retail websites assisting customers with their inquiries about the products and services. Financial chat bots provide support with respect to queries relating to current account, credit cards, depute transactions etc. Recently there has been an immense growth in conversational agents, Web-based agents offered scalability which gave access to a massive amount of population through internet. Additionally, computation linguistic have undergone some advancement in parsing technologies boosting natural understanding capabilities. Popularity of these conversational agents are ever increasing, as the industry is demanding cost effective solutions for handling the queries of the customer. They have been proven in extremely significant in providing customer support and indeed saving a whole number of resources and training time compared to the traditional industry procedures (Lester, Branting and Mott, 2004).

2.1.1 Rule - Based Systems

Rule - based chatbot gives out the desired output by considering the rules defined. These methods handle the queries of the user a lot quicker compared to traditional chatbots. The conversations flow like a dropdown menu, which lets user select the desired option with a click a button. They are usually considered when we must perform simple tasks like creating tickets and scheduling calls (Leah, 2022).

ELIZA was the first chatbot system developed in 1966 by Joseph Weizenbaum. It is a natural language processing computer program which shares a similar behaviour to therapist, transforming the statements to relevant questions by the user. Conversations between the user and program progresses combining pattern matching and substitution methodology (Sharma, et al, 2017). The following example is the most famous ELIZA conversational, excerpted from (Mandasari, 2019).

Algorithm 1 Excerpt of ELIZA algorithm (Mandasari, 2019)

```
function ELIZA GENERATOR (user sentence) return response.
                Find the word w in sentence that has the highest keyword rank
    if w exists then
        Choose the highest ranked rule r for w that matches sentence
        response ← Apply the transform in r to sentence
        if w = 'my' then
            future ← Apply a transformation to sentence
            Push future onto memory stack
        end if
    else                                              (no keyword applies)
        either
        response ← Apply the transform for the NONE keyword to sentence
        or
        response ← Pop the top response from the memory stack
    end if
    return response
end function
```

If we take the example of another rule-based chatbot ALICE (Artificial Linguistic Internet computer Entity) which uses AIML (Artificial Intelligence Mark-Up Language) to output response with respect to inputs by the users. The generation of the response is based on Natural language understanding and pattern matching. It consists of mainly two components namely "chatbot engine" and "language model" which makes implementation easy in a newly developed knowledge model. AIML files contain the language model. AIML's primary design feature is minimalism, and it is possibly the simplest of all chat robot languages. As previously stated, the fundamental unit of knowledge in AIML is the category. Each category includes an input or question, an output or answer, and an optional context. The pattern refers to the question. The template is the answer or response. The two types of optional context are "that" and "topic." When working with AIML, pattern matching is very simple because it only consists of words, spaces, and wildcard symbols and *(Sharma et al., 2017).

2.1.2 Corpus – Based Systems

Corpus-based chatbot are developed to overcome the in efficiencies in the rule-based systems. Although the template-based chatbots are quick in terms of adoption and deployment but it fails to performance when the number of stored pattern-template pairs increases. The text files are in the AIML format which requires the system to search word by

word through all files to match template. Due to this reason the inclination changed towards the corpus-based system, which no longer use patten matching method. Previously, for the storage and the receival of information we were using databases. In 2007 a method was proposed that let the identification of the relevant attributes and values from the user messages, which therefore creates the subsystem that queries the database for responses. An alternative method known as ontology model or semantic web is used for storing domain knowledge. Unlike before this method utilises semantic query language (SPARQL) to manipulate the semantic web. A second type of corpus-based chatbots that works on the technique of word embeddings, could retrieve the correct responses as the query-response pairs was stored in the form of vectors. They output responses based on the distances between the user input and query-response pairs. Additionally, there are several advantages using corpus-based chatbots over the rule-based. It doesn't require language tags and wildcard characters thus simplifying creation. The separate response retrieval enhances the quality of responses and matching letting chatbot to output more accurate and specific information compared to before (Luo et al., 2022).

2.2 Architecture of a chatbot

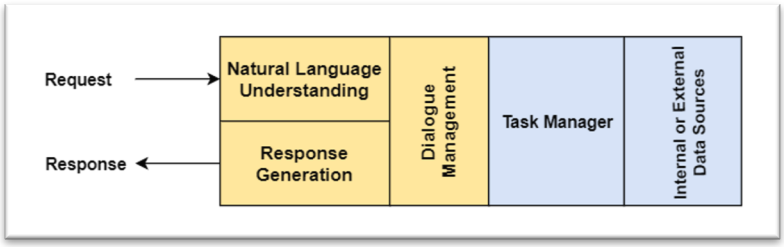

Figure 1 Architecture of Chatbot (Bapat, 2017)

Starting with the user request which is interpreted by the natural language understanding component (NLU) that makes an inference of the user intentions. As soon as the user input is known further action is taken. Dialogue management is responsible for finding out the continuation of the topic. Chatbot can be of various types namely conversational, informational, and transactional. The chatbot retrieves the data from the data sources in the case of informational and conversational. If the chatbot is built for transactional purposes, then further actions are taken by task manager. Task manager performs the requested action and return to the user. The above figure.1 illustrates the high-level architecture of chatbot. We will expand of the components further below (Bapat, 2017).

2.2.1 Natural Language Understanding

The natural language understanding component processes the request by the user, the component parse the inputted statement and tries to extract the intent after the understanding the user's intention and information associated. To get structured data from the users Pre-processing is done by utilising natural language techniques. The technique is tokenization, which dissembles the sentence to form a series of tokens. For examples 'What are you wearing?' tokens are given as: 'What' 'are' 'you' 'wearing' '?'. Other technique is lemmatization, which is like tokenization, just the difference between both is the lemmatization accurately does the reduction to root words. Stemming strips, the sentences by patten matching. Lastly, user entity is indemnified and extracted which combines people, places, companies, etc. Finally, the structured data is ready to be fed natural language understanding strategies for getting meaning out of it. Strategies used in chatbots are:

- **Rule- Based Pattern Matching**

  To categorise the text and compare user input to predefined patterns in this situation, chatbots use pattern matching. For pattern matching chatbots utilise Artificial Intelligence Mark-up Language (AIML). The pattern is matched is based on the predefined user template and every pattern hold a response. Different methods can be selected according to the requirements and scope (Wallace, 2003.)

- **Machine Learning**

  To correctly understand the user, the NLU unit attempts to parse the user requests and categorises them. Intents is referred to link between the intentions of user and the particular action associated with it. Entities are natural language phrases which are based on the selected intent. Actions are defined as the output that is given by the chatbot when the particular entity is processed. An action has parameters that allow you to specify more specific information about it.

2.2.2 Dialogue Manager

To fulfil the user's request, the conversation manager coordinates a natural language dialogue with the user and works with services like an inference engine, knowledge base, planner, and

external business services. Text and speech may be supported interaction modalities (Telang et al., 2018). Dialogue management is defined as the functions which are performed by the dialogue manager, these functions are updating of the context of conversation, providing context for interpretation of observed signals, coordinating components and take context related decisions (Traum and Larsson, 2003). Dialogue management strategies are discussed below:

- **Finite - state dialogue management**

  In this technique, the conversational states are predefined, and the dialogue manager maintains the conversational interaction's status in accordance with these states. The chatbot can precisely identify which state the conversational statement is in thanks to its finite state sequence. This is the most straightforward approach to creating a dialogue manager (QUARTERONI and MANANDHAR, 2009).

- **Frame - based or Form based dialogue management**

  The system requires the user to follow the defined path in the scenario of the query and users have limited alternatives to select from. The system won't be able to handle any queries by the user which are deviated from the pre-defined path. On successful events for a free- based dialogue, returning and changing of the responses are extremely difficult for the user resulting dialogue to divert to different branch of the tree (Goddeau et al., 1996).

- **Initiative-based dialogue management**

  The information state of a dialogue differentiates information from all the available information. The continuing system response depends on the information state of dialogue which contains user interventions and previous dialog actions, and this state is also known as conversational store, discourse context, or mental state (Griol et al., 2014).

2.2.3 Task Manager

Task manager performs a series of steps which is required to obtain a desired action. If we talk about transactional chatbot, task manager is responsible for executing the tasks utilising the internal and external data sources. A perfect example is the train booking application. In the case of informational chatbot, it acts as an information retrieval manager. Task managers retrieve data from database or templates in the case of conversational chatbots (Bapat, 2017).

2.2.4 Data Sources

After processing the request by the user, the intents are being matched with the user's intentions. Additionally, the chatbot handle the queries by receiving the data from data sources or perform the necessary actions. The data can be fetched from various resources mainly from the own knowledge base or externally through an calling an API (Adamopoulou and Moussiades, 2020). For example, chatbots allow the user to request for the weather update, the chatbot will call the third party whether API for completing that request.

2.2.5 Response Generation

For a communication between the user and the chatbot, the conversational agent must return a reply. The reply given by the agent should be aligned with the conversation context. This is done by the combined effect of the two different modules. The first modules give us the list of replies and the other ranks them and select the one with the highest score. There are two response generation models discuss below:

Retrieval based models

These models offer a greater flexibility, utilises the predefined responses to user's request. For response creation resources are accessed using API's (Hien et al., 2018). Before using the matching strategy for response selection, a retrieval-based chatbot retrieves some response candidates from an index (Adamopoulou and Moussiades, 2020)

Generative models

Textual outputs by the conversational agents are a sequential task. Recurrent Neural Networks (RNN) are perfect in terms of handling these sequential tasks. What makes them powerful is the high-dimensional hidden state with non-linear dynamics which gives them an

advantage of processing and memorising information faster than before. Moreover, computation of these gradients is cheap with backpropagation through time. Although with several advantages there are disadvantages in the Recurrent Neural Networks, the training of such networks is difficult due to the instability in the relationship between the parameters and the dynamics of the hidden state which eventually result in the problem of gradient vanishing and gradient exploding which result in exponential increase or decrease in training time of the network (Sutskever, Martens and Hinton, 2011). This problem is furtherly explained in (Roodschild, Gotay Sardiñas, and Will, 2020), the shallower layers doesn't experience the weight update as expected. This phenomenon gets worse as hidden layers increases. For correcting this issue many suggestions are given out that included altering the activation function. Although, it's a compulsory to include sigmoid activation function for several architectures that include long short-term memory (LSTM), gated recurrent unit (GRU) and autoencoders. Later we have seen deep belief networks which are based on layer-by-layer training. Lastly, we had one technique; batch normalization which including some additional trainable parameters but handled this problem efficiently.

2.3 Sentence and word embedding

Word Embedding are the vector representations of the words which is then given to the neural network. There are several techniques available which can help us in converting the words into numerical representations namely, Word2vec (Mikolov et al, 2013), GloVe (Pennington, Socher and Manning, 2014), etc. The probability of occurrence of words is predicted after being trained in the neural network. Semantic vector models assign the words with real values and hold the features to support a variety of applications like information retrieval, document classification, question answering, named entity recognition and parsing. An additional understanding of the vectors can be achieved by adding them (Mikolov et al, 2013).

For information retrieval cosine similarity metric is utilised in amount. This can be used to find the similarity between the two documents by calculating cosine values. On user request the highest similarity score between query term vector and document term vector represents the relevancy of the query and document (Rahutomo, Kitasuka and Aritsugi 2012).

2.4 Fine tuning and Transfer Learning

This concept related to utilisation of the learning from a pre-trained model for a specific task to our model for solving a similar task. The only prerequisite in transfer learning is that there should be a connection between the learning activities. Fig. 2 displays us a perfect example of transfer learning if a person is an expert in playing the violin and looking forward to learning piano in future. He will be able to learn piano a lot faster as compared to the people who never had access to learn violin.

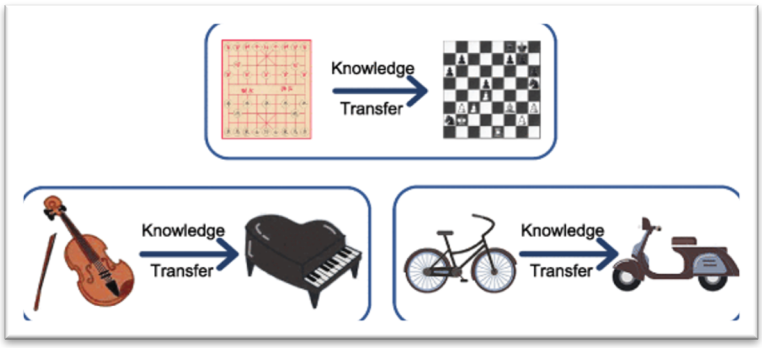

Figure 2. Transfer Learning (Zhuang et al., 2020).

It is important to note that the concept of transfer learning doesn't result positive in some new tasks and therefore the leaning can fail. For example, the learning to ride a bicycle won't be making any advantage in learning piano. Additionally, it is not a necessity that the transfer will facilitate in similar domains, if we take an example to understand learning Spanish and French belong to the same Romance group of languages but learning both is not interdepended. There may be clash in word formation, usage, pronunciation, conjugation due the concept called negative learning (Zhuang et al., 2020).

Fine-tuning is beneficial for several reasons. First is the combination of the labelled data and unlabelled training data. Massive amount of unlabelled data can be discovered with combing small amount of labelled data. It is always preferred to consider small amount of labelled data for achieving high performance as its discovery is difficult. Additionally, fine-tuning provides a solution to the mismatches between the training data and the population of interest. The test data is developed very similar to the training data, and it is expected by the user to want something else which is not in training data, in that case PTMs need to update with the world (Church, Chen and Ma, 2021).

2.5 Seq2Seq or encoder decoder

The Sequence-to-sequence (Seq2Seq) also known as Encoder-Decoder model which uses a multi-layered Long Short-Term Memory (LSTM) for the mapping of input sequence to vector equivalent (Sutskever, Vinyals and Le, 2014). LSTM is a two-component model, first the input sequence is converted into low dimensional representation then it is furtherly converted back to its original form. The blocks of Encoder-decoder model are vanilla RNN or its variations. The words generation are dependent on model and preceding generated word. EOS (end-of-sentence) tokens are combined at the last positions of every sentence (Prakash, 2016).

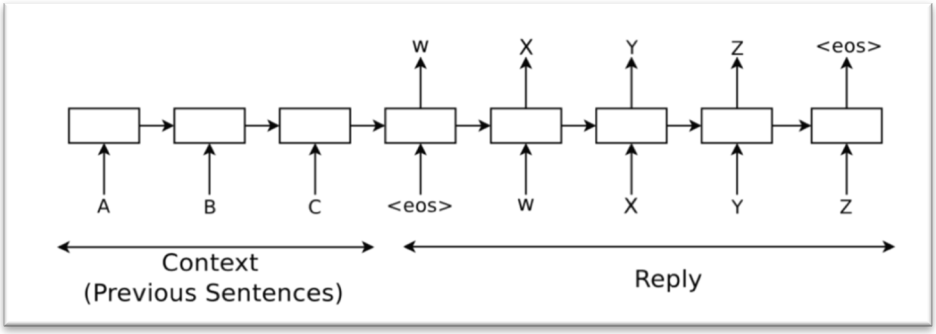

Figure 3. Sequence to Sequence Framework (Mathur and Singh, 2018).

For building a conversational agent sequence-to-sequence and hierarchical approaches are taken as the baseline models. Additionally, there are being continuous attempts to make these agents more robust. Large amount of dialogue data is required due to their huge parameter space. A weighting model is introduced for overcoming the limitation; the network tends to learn the response without the knowledge of speaker identification (Mathur and Singh, 2018).

2.6 Joint Intent Classification and Named Entity Recognition

Recently there has been several developments related to training intent classification and named entity recognition (NER) in a multi-task setup. Taking the idea forward that both the intent and semantic slots of a sentence are correlative (Zhang and Wang, 2016) presented a joint Bidirectional Gated Recurrent Unit (BiGRU) model that could provide the solution both tasks, where GRU is responsible for learning the representation of each time step for predicting the label of each slot. Additionally, intent classification is done by capturing the global features with the help of max-pooling layer. (Chen, Zhuo and Wang, 2019) used state

of the art model; BERT (Bidirectional Encoder Representations from Transformers) for joint intent classification and slot filling and are able to achieve an inclination in the accuracy of intent classification and sentence-level semantic frame. Moreover, an increase F1 score with regards to slot filling. For the creation of a direct connection between the two tasks that could promote each other (Niu, Chen and Song, 2019) proposed a co-attention network additional to the intent and entity attention units. (Vanzo, Bastianelli and Lemon, 2019) proposed a hierarchical multi-task architecture which includes a self-attention mechanism and BiLSTM encoders with CRF tagging layers. Lastly, the work given by (Baker, Fillmore and Lowe, 1998) predicated dialogue acts, intents and entity labels.

2.7 Open-Source Conversational AI, RASA

Rasa open source, a machine leaning platform which provides the suitable tools and infrastructure for building conversational agents. They provide the flexibility to automate conversation between computer and human which can be integrated to websites and social media platform. Rasa open source comprises of mainly two components; the first is NLU (Natural Language Understanding), which is responsible for parsing the intent and entity of the user and Rasa Core performs the best constituting response or the actions required remembering the previous conversation. Additionally, Rasa X which is a free closed source toolkit allows continuous integration and continuous deployment (CI/CD). Moreover, Rasa let the user to deploy their application on various platforms namely, WhatsApp, Telegram, Slack, Facebook Messenger, etc (Introduction to Rasa Open Source, 2022).

2.7.1 Rasa NLU

NLU stands for natural language understanding, its function is to convert user inputs to structured data. The NLU lets developer to define the combination of the inputs with respect to the intents. Below is example of sample NLU data.

Sample NLU data

intent: Bye

- Goodbye
- Bye

- See you again next time.
- Bye, see you soon.

As soon as user inputs the message, the NLU component able to classify the intent and entity with respect to the response. For example, when the user messages “Goodbye”, then the associated intent “Bye” is classified. Entity is taken into consideration from the sentences. For example, user messages “My name is Lakshay”, the chatbot will take out the name entity; “Lakshay” and will remember thought out the conversation.

The input messages by the user follows through a of pipeline. The pipeline contains mainly three components:

- Tokenization
- Featurization
- Entity Recognition, Intent Classification and Response Selectors.

Tokenization

The tokenizer divides the input messages by the user into a sequence of tokens. There are various types of tokenizer present in Rasa. For example, Whitespace Tokenizer divide the words based on the spaces. Other tokenizer can be utilised according to the sentences which are not whitespace tokenized.

Featurization

A featurizer creates the vector representations of the user input sequence. There are two different types of featurizer present namely, sparse featurizer and dense featurizer. The sparse featurizer uses a huge memory when it converts to the vector equivalent, and the returned vector contains number of missing values. These features store the values which are non-zero and their position in the vector (Components, 2022).

Entity Recognition, Intent Classification and Response Selector

Entity extractors mainly gives out the entity in the user message, the following entities can vary through person name, location, etc. Intent classifiers match the intent in the user message to the already defined domain file. Response selector is associated to predict a response from the chatbot.

2.7.2 Rasa Core

Rasa Core manages the dialogue management, is it responsible for the committing conversation and the response flow. The defined path that the chatbot follows is given in the *stories.* An instance contains the user's intent and the response associated with that intent. The output given out by the chatbot are based on the templates called *responses*. Then comes the *domain*, where the bot operates from. The domain combines all the intents, entities and actions used by the agent is given. Additionally, we can define memory *slots,* which retains the information of the user for example, First Name, Last Name, etc.

Story

Training our dialogue management model requires stories to be defined in Rasa. Stories represents the defined path, which is used by the agents when communicating with the user. These paths include the user's intent, entity and the actions that needs to be performed Below we have given an example to story (Introduction to Rasa Open Source, 2022).

Sample Story used in training

```
story: path 11
    steps:
    - intent: greet
    - action: utter_greet
    - intent: first_name
      entities:
        - firstname: "firstname"
    - action: action_save_firstname
    - intent: last_name
      entities:
        - lastname: "lastname"
    - action: action_save_lastname
    - action: utter_to_ask
    - intent: storedatabase
    - action: action_store
    - action: utter_intro
    - action: utter_choice
    - intent: options_account
    - action: utter_Account_options
    - intent: options_Cards
    - action: utter_cards
```

Domain

The domain provides our assistant with the platform to operate. It let us specifies the intents, entities that our assistant should know, additionally the responses and the actions that is utilised by the agent. Moreover, we can define buttons in domains which makes convenient for the user to query quickly (Introduction to Rasa Open Source, 2022).

A sample domain file from training data

```
intents:
- online_card_payment
- money_limit
- difference_available_ledger_balance
- pending_and_upcoming_transactions
- Direct_debits

entities:
- firstname
- lastname
```

Slot

Our agent contains memory slots; key-value database. A slot retains the information provided by the user during the conversation, the information could be anything from personal name or a location. Slots provide the flexibility to retrieve information from various data sources.

Supported slot types

Text
Boolean
Categorical
Float
List
Unfeaturized

A sample example from training data

```
slots:
  firstname:
    type: text
    influence_conversation: true
    mapping:
    - type: from_entity
      entity: firstname
 lastname:
    type: text
    influence_conversation: true
```

```
    mapping:
    - type: from_entity
      entity: lastname
```

Response

Responses are the output given by our assistant when the user queries, the responses usually consist of text, images, and buttons (Introduction to Rasa Open Source, 2022).

A sample domain file from training data

```
responses:
  utter_greet:
  - text: Hi I’m Lakshay, your virtual assistant. I help customers like you with their everyday banking queries. Please provide your first name and last name to get started?
  utter_to_ask:
  - text: Would you like me to save your first name and last name?
```

Chapter 3: Design and Development

This chapter will include the design and development of our financial customer support chatbot. Particularly, it will combine data collection, design and analysis, creation of NLU data, training NLU model, dialogue management and integration are the focus areas.

3.1 Chatbot Architecture

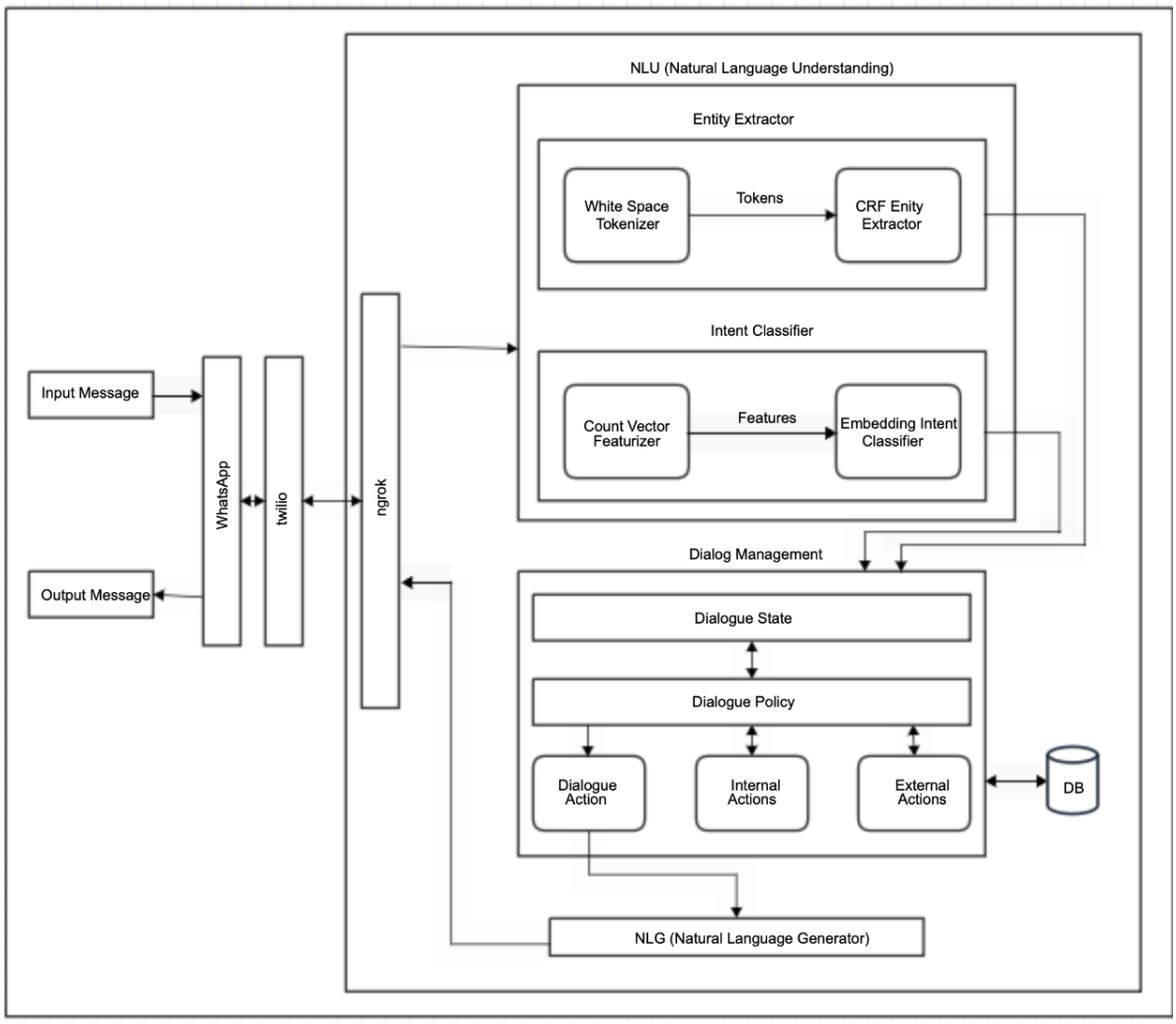

Figure 4. Architecture of banking customer support chatbot

This work is based on chatbot which is built on the Rasa open-source Framework, with an objective of aiding the customers having banking queries. Fig 5. displays us the architecture of our chatbot.

3.1.1 Natural language Understanding Unit

On the request by the user, the sequence message follows through the NLU (Natural Language Understanding Unit). This unit is responsible for performing mainly two tasks which includes entity extraction and intent classification.

3.1.1.1 Entity Extractor

As soon as the NLU unit receive the user's input, it goes to the entity extractor which tokenize the sequence for further processing and understanding. The tokenization happens with the help of WhitespaceTokenizer (Smarr et al., n.d.) which splits the message on white spaces between the words of the sentence. It gives a flexibility to user to treat EOL characters as whitespace or as a token. The generated tokens then go to the next component where the entities are extracted from the sequence of the words. For extracting the entities, we have used CRFEntityExtractor which implements conditional random fields (CRF) to perform named entity recognition (Introduction to Rasa Open Source, 2022). The NER works by selecting a word for which it searches the features from the surrounding words. We receive output as entities with their associated labels and confidence matrix of the predictions.

3.1.1.2 Intent Classifier

The next step is finding out the intent from the user message. For intent classification there are two components further which does the work. Featurizer help the classification model to learn pattens out of the tokens. Specifically, we have used CountVectorsFeaturizer which creates features for intent classification and response selection by converting the user input into tokens using bag-of-words representation (Introduction to Rasa Open Source, 2022).

On successfully retrieving the features they are then used for training the intent classification model. For classifying we have used EmbeddingIntentClassifier which performs the embedding of user message and intent labels into same space. Additionally, the training is done keeping the highest similarity between embeddings and intent prediction.

3.1.2 Dialog Management

On the retrieval of the entities and intent from the previous components, it is now fed to the dialogue management unit which comprises of further three components namely Dialog state, Dialog Policy and Actions.

3.1.2.1 Dialog Policy

Dialog policy is responsible for deciding mainly two things, how our dialog system will response and what the response would be during conversation. We have utilised three policies that helped in the development of this banking customer support chatbot.

Memorization policy remembers the stories from the training data. It matches the conversation to the pre-defined stories in the stories.yml file. Additionally, it offers an alterable parameter max_history which was set to 5 in our case, the memorization policy will remember last 5 policy when matching with the training data. Once it finds a matching fragment, the next action is predicted with the confidence of 1.0 (Policies, 2022).

The Rule Policy drives the conversation according to the defined rules in the rules.yml file. For example, from our chatbot we have define a rule that stats as soon as the user say “Bye”, the chatbot respond with a goodbye message. Basically, rules are defined to handle any unexpected user inputs that would be difficult for chatbot to handle (Policies, 2022).

The Transformer Embedding Dialogue (TED) policy is used for predicting the following action and the entity. The TED architecture comprises of transformer encoder. The prediction of entity labels is done through conditional random field (CRF) tagging layer which is placed on the top transformer encoder output. The next action is predicted through embedding the encoder output and action labels into a single semantic vector space. We have used TED policy with max_history parameter set to 5, for predicting next response it will consider the last 5 fragments in the conversation and epoch: 100 (Policies, 2022).

3.1.2.2 Actions and Natural Language Generator

Next comes the actions which will trigger processes following it. Actions are divided into two categories which are external actions and internal actions. Internal actions combine all those responses which the system response internally. Although external actions are which the system request external sources to fulfil that request. For example, when the user sends the request the information regarding the books available in data science then for handling the user request the system will retrieve data from external database and great example is requesting an API for whether information.

The result then passes through Natural Language Generator which will pass on the response further to the global server for displays the output to the user.

3.2 Chatbot Design and Implementation

3.2.1 Data collecting Approach

To make a start, we have taken all the frequently asked questions and first-time account holder queries that are available publicly on the HSBC UK website (www.hsbc.co.uk, n.d.). Additionally, we have created a chatbot that shares the functionality to the chatbot been used by the HSBC Group; MOBA.
Some of the top queries included in our chatbot are:
Why has my online card payment been declined?
How much money can I send via the app?
What is the difference between my available balance and ledge balance?
What is included in my pending and upcoming transactions?
How do I manage my Direct Debits?
How do I manage my standing orders?
How do I change my contact details?
What happens if I don't recognise or want to dispute a card transaction?
Where can I find my IBAN/BIC?
How do I find my nearest 'free to use' cash machine?
Additionally, we have considered the data which relates to banking, borrowing, investing, insurance, wellbeing in the development of our banking chatbot.

3.2.2 Designing NLU unit

Designing Natural Language Understanding is extremely important task when it comes to handling the user queries. It can be defined in the nlu.yml file present in the rasa framework. NLU unit gives the flexibility to the user in querying in multiple ways, without worrying about the using the specific sentence format for querying. In other words, we can give different examples how a user can ask something to our bot for a particular intent. Labelling of the entities is another crucial factor when it comes to the efficiency in the response. It is necessary for the chatbot to reply accurately on the condition of entity matching. If the system can retrieve the entity from the user input our chatbot will output as expected.

nlu.yml

```
200
201  - intent: Loans
202    examples: |
203      - Loans
204      - Apply for loans
205      - Personal loan
206      - i want to loan
207      - i want to apply for loan
208      - Motor vehicle loan
209      - Motor vehicle
210      - i want to apply for Motor vehicle loan
211      - Debt consolidation loan
212      - Debt consolidation
213      - i want to apply for Debt consolidation loan
214      - Debt consolidation from diffirent bank loan
215      - i want to apply for Debt consolidation from diffirent bank
216      - Home improvements loan
217      - Home improvements
218      - i want to apply for Home improvements loan
219      - Holidays loan
220      - Holidays
221      - i want to apply for Holidays loan
222      - Furniture and fittings loan
223      - Furniture and fittings
224      - i want to apply for Furniture and fittings loan
225      - Annual commitments loan
226      - Annual commitments
227      - i want to apply for other needs loan
228      - other needs loan
229      - other needs
230      - i want to apply for Annual commitments loan
231
232  - intent: Overdrafts
233    examples: |
234      - How an overdrafts works
235      - Will using or applying for an overdraft affect my credit rating?
236      - What's the difference between an arranged and an unarranged overdreft?
237      - Are there any fees for using an overdraft?
238      - How much does an unarranged overdraft cost?
```

Figure 5. Designing intents

The figure above displays us the intents and their associated examples. Loans, Overdrafts are the intents. Loans intent consists of various loan types be it personal loans, motor vehicle, debt consolidation, home improvements, holidays, furniture and fittings, annual commitments, other need loans, etc. All these intents are a part of a main intent that is loans.

Extracting entity has a benefit when it comes to giving out a response. For example, when the user query “Apply for loans” intent of this sentence is Loans. If our chatbot won’t consider entity the chatbot will have to again request user, the specific loan they want to go with and will return all the information about the loans available.

nlu.yml

```
86
87  - intent: greet
88    examples: |
89      - hey
90      - hello
91      - hi
92      - hello there
93      - good morning
94      - good evening
95      - moin
96      - hey there
97      - let's go
98      - hey dude
99      - goodmorning
100     - goodevening
101     - good afternoon
102
103 - intent: goodbye
104   examples: |
105     - cu
106     - good by
107     - cee you later
108     - good night
109     - bye
110     - goodbye
111     - have a nice day
112     - see you around
113     - bye bye
114     - see you later
115
116 - intent: affirm
117   examples: |
118     - yes
119     - y
120     - indeed
121     - of course
122     - that sounds good
123     - correct
124
```

Figure 6. Designing acronyms

For chatbot to understand greet, goodbye and affirms from the user which can be alternated depending on the user, we must give model combination of inputs. Additionally, it is

important to give multiple spelling of words which can be written separately or jointly, still mean the same. Such pattens should be train by the model for handling such irregularities.

From picture above we can see that various combination of good morning has been defined to handle the above discussed situation. Similarly list of acronyms for each of the intent have been defined for the chatbot to train accordingly.

3.2.3 Training Stories Design

Training our dialogue management model requires stories to be defined in Rasa. Stories represents the defined path, which is used by the agents when communicating with the user. These paths include the user's intent, entity and the actions that needs to be performed (Introduction to Rasa Open Source, 2022).

stories.yml

```yaml
version: "3.0"

stories:

- story: path11
  steps:
  - intent: greet
  - action: utter_greet
  - intent: first_name
    entities:
    - firstname: "firstname"
  - action: action_save_firstname
  - intent: last_name
    entities:
    - lastname: "lastname"
  - action: action_save_lastname
  - action: utter_to_ask
  - intent: storedatabase
  - action: action_store
  - action: utter_intro
  - action: utter_choice
  - intent: options_account
  - action: utter_Account_options
  - intent: options_Cards
  - action: utter_cards
```

Figure 7. Designing stories

Figure 7 shows the story path 11, in this query the user wants to know about the Cards offering. The query starts from greeting the user, then the chatbot request user to input first

name and last name, which is then stored in the database. The system prompts account options to select from for further information to select from.

stories.yml

```
118
119  - story: path16_product
120    steps:
121    - intent: greet
122    - action: utter_greet
123    - intent: first_name
124      entities:
125      - firstname: "firstname"
126    - action: action_save_firstname
127    - action: utter_LName
128    - intent: last_name
129      entities:
130      - lastname: "lastname"
131    - action: action_save_lastname
132    - intent: storedatabase
133    - action: action_store
134    - action: utter_intro
135    - action: utter_choice
136    - intent: options_account
137    - action: utter_options_account
138    - intent: options_Different_query
139    - action: utter_choice
140    - intent: options_Product_Lending
141    - action: utter_Product_options
142    - intent: options_savings
143    - action: utter_Savings
144
```

Figure 8. Designing stories

The figure 8 shows the story, which lets the user to back propagate to the main account options which gives the user the flexibility to select different query. The query starts the same as above story till saving the information of the user first name and last name then allowing them to select a category from their account.

3.2.4 Domain Design

The domain provides our assistant with the platform to operate. It let us specifies the intents, entities that our assistant should know, additionally the responses and the actions that is utilised by the agent. (Introduction to Rasa Open Source, 2022). The domain file consists of intents, entities, slots, actions, buttons, and the responses. All these are used by our chatbot while processing the user request.

The responses contain utter which will help our chatbot with communicating with the user. Additionally, the utter can have simple text or buttons. The buttons in the response allow the user to get the response a lot quicker compared to the default setting. For executing of the actions associated with response we must give actions in the domain file for the chatbot to use.

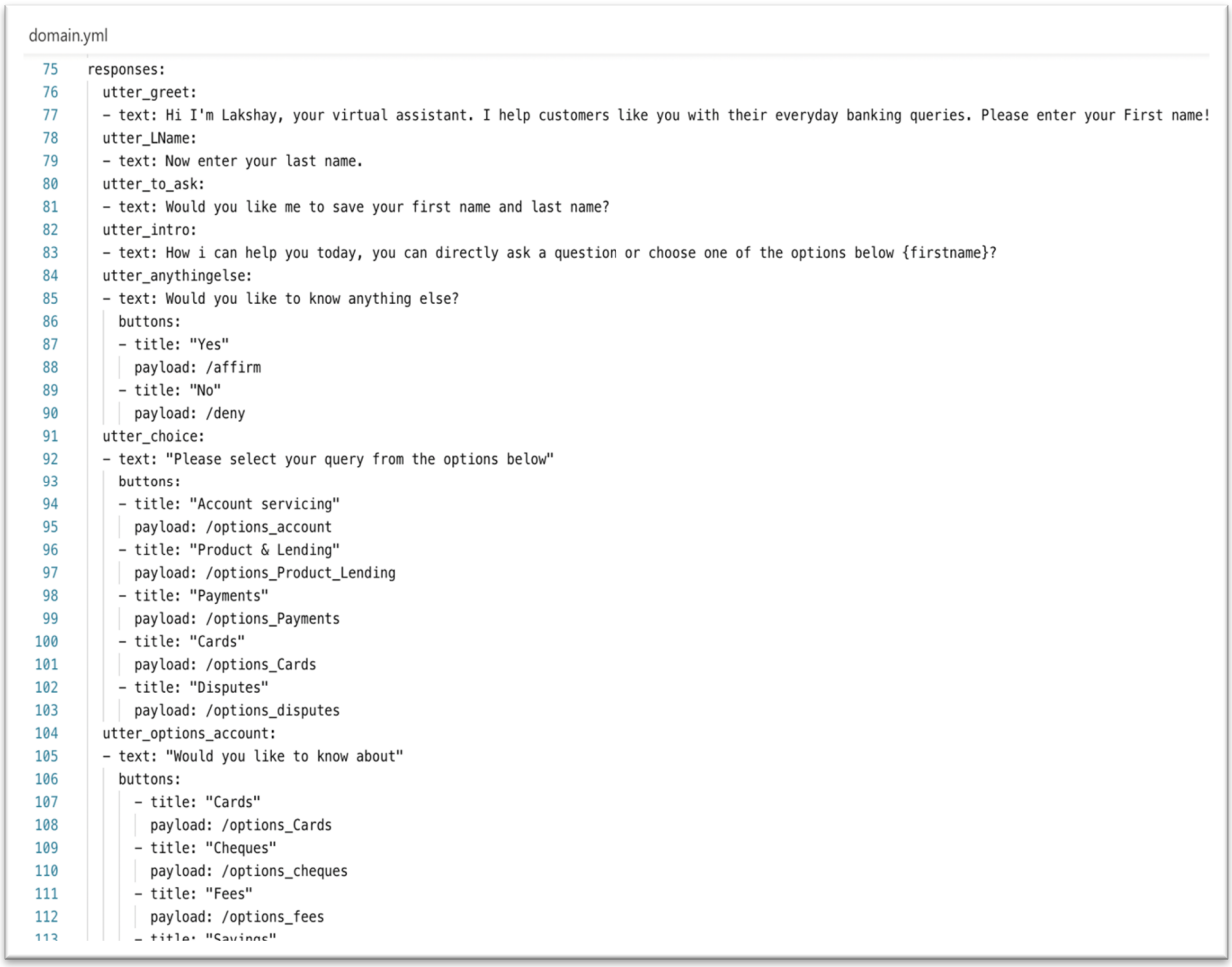

Figure 9. Designing responses

Figure 9 shows the responses that our chatbot uses for having a conversation. For example, utter_greet greet's the user by displaying "Hi I'm Lakshay, your virtual assistant. I help customers like you with their everyday banking queries". Moreover, we have utilised buttons in utter_choice, utter_options_account, utter_anythingelse. Using buttons displays user several options to choose from without expecting some additional input from the user which results in the handling the query a lot quicker.

```
239     utter_thanks:
240     - text: You're welcome.
241     utter_goodbye:
242     - text: Bye!
243     utter_deny:
244     - text: Thankyou for your time!
245     utter_iamabot:
246     - text: I am a bot, powered by Rasa.
247
248   actions:
249   - action_save_firstname
250   - action_save_lastname
251   - action_store
252   - utter_greet
253   - utter_choice
254   - options_Different_query
255
```

Figure 10. Designing actions

Additionally, as shown in the figure 10 we need to the give all the actions that we want our chatbot to perform during the conversation.

domain.yml

```
51  - options_Cards_AC
52  - options_Cards_Pin
53  - options_Personal_loan
54  - options_Different_query
55  - options_Credit_Cards
56  entities:
57
58  - firstname
59  - lastname
60
61  slots:
62    firstname:
63      type: text
64      influence_conversation: true
65      mappings:
66      - type: from_entity
67        entity: firstname
68    lastname:
69      type: text
70      influence_conversation: true
71      mappings:
72      - type: from_entity
73        entity: lastname
74
75  responses:
76    utter_greet:
77    - text: Hi I'm Lakshay, your virtual assistant. I help customers like you with their everyday banking queries. Please enter your First name!
78    utter_LName:
79    - text: Now enter your last name.
80    utter_to_ask:
81    - text: Would you like me to save your first name and last name?
```

Figure 11. Designing slots

Lastly, from figure 11 we have utilised slots in the domain file. They hold the information of the user as a key-value store. We have used them to store the first name and the last name of the user. Slots provide the parameter influence_conversation which is set to true, which means the chatbot will remember the input value from the user during the next fragments of

the conversation. And the mapping has been used for chatbot to know the entity; the response expected from the user to input.

3.2.5 Custom Actions

Custom action in rasa framework is used for running any code depending on the requirements. This can be used for making API calls or querying something from an external database (Actions, 2022). From the figure 12 we have defined actions to store the information of the user into the database. For storing the first name of the user we have made class Actionsavefirstname (Actions) which retrieve the first name of the user. This action takes the value from the slots that have been previously defined. Similarly, for storing the last name of user, we have made class Actionsavelastname (Actions) which performs the same for storing last name.

actions.py

```
from typing import Any, Text, Dict, List
from rasa_sdk import Action, Tracker
from rasa_sdk.events import SlotSet
from rasa_sdk.executor import CollectingDispatcher
import requests
import sqlite3
import speech_recognition as sr
from translate import Translator


class Actionsavefirstname(Action):

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

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        print(tracker.get_slot('firstname'))
        return [SlotSet('firstname',tracker.latest_message['text'])]

class Actionsavelastname(Action):

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

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

        dispatcher.utter_template("utter_to_ask", tracker)
        print(tracker.get_slot('lastname'))

        return [SlotSet('lastname',tracker.latest_message['text'])]
```

Figure 12. Designing custom actions

actions.py

```
44
45
46  def datastore(firstname, lastname):
47      conn=sqlite3.connect('cov.db')
48      mycursor = conn.cursor()
49      mycursor.execute("""CREATE TABLE IF NOT EXISTS my_info (Name TEXT, HomeState TEXT);""")
50      mycursor.execute("INSERT INTO my_info VALUES (?,?)",(firstname,lastname))
51      conn.commit()
52      print(mycursor.rowcount,"record inserted")
53
54
55  class ActionStore(Action):
56
57      def name(self) -> Text:
58          return "action_store"
59
60      def run(self, dispatcher: CollectingDispatcher,
61              tracker: Tracker,
62              domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
63          x=tracker.get_slot('firstname')
64          y=tracker.get_slot('lastname')
65          datastore(x,y)
66          dispatcher.utter_message("Thank you! Your information is saved.")
67          print(x)
68          print(y)
69          return []
70
71
72
```

Figure 13. Database connection

After the actions are made, we now must make a connection to the database. For doing the same we have defined a function def datastore (firstname, lastname): which successfully connect with our slqlite3 database we are using in this development. First the query checks if the table exists or not, then make one if it doesn't find one. The next query inserted the user's first name and last name to the table as given in figure 13.

Additionally, class ActionStore (Action): is defined in figure 13 for storing the previously received values from the user. As soon as the user data is stored in the database a message is displays to the user saying "Thank you! Your information is saved".

3.2.6 Pipeline Design

For designing the pipeline used in our development of banking customer support chatbot I compared the DIET model and a SpacyNLP model.

The RASA framework provides the flexibility to choose various pipelines without writing lines of code and can be defined in the config.yml. The reason for choosing the DIET configurations as our chatbot is domain specific (banking), which offers extremely capable of identifying the intents and entities in the user messages. Additionally, we have utilised SpacyNLP which is pre-trained model on multiple types of textual data for example texts, comments, and blogs and number of policies for the prediction of next action.

3.2.6.1 The DIET Model

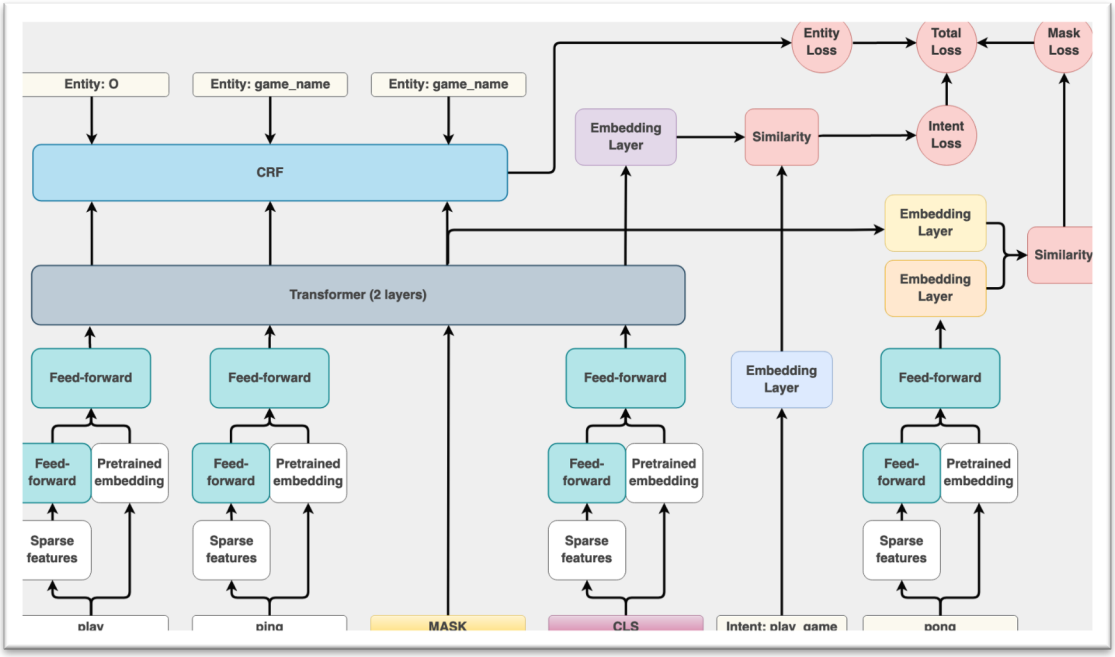

Figure 14. DIET Architecture (Bunk, 2020)

Featurization

The sentences are brokens into a series of words or subworlds called tokens. We add the __CLS__ token at the end of sentences to differentiate from the whole document. Then the input tokens are converted to numerical representations using sparse and dense features which are basically one-hot encoding and multi-level encodings of character n-grams. As the sparse features contain information, which is not relevant, so we tend to dropout these to avoid a condition of overfitting. In the case of ConveRT we take the sentence encoding as __CLS__ because doing that combines extra contextual information with the information

gained from words embeddings. For the out of the box pre trained BERT the output is set to mean of the embeddings for the BERT [CLS] token and for GloVe. For matching the dimension of the dense features, we let sparse features through the fully connected layer (Bunk, 2020).

Transformer

Transformer was first introduced in the research paper "Attention Is All You Need" (Vaswani, 2017). For encoding the contextual information in the sentences, we have utilised a 2 layered transformer with relative position attention. It is required for the transformer architecture to receive the same dimensional input as the transformer layers. Therefore, for obtaining the same the concatenated features are allowed to go through other fully connected layer with same weights (Bunk, 2020).

Named entity recognition

Prediction of Entity labels y_{entity} happens with a Conditional Random Field (CRF) which is used for sequence tagging, this layer in integrated above the transformer out sequence (Bunk, 2020).

$$L_E = L_{CRF}(\boldsymbol{a}, \boldsymbol{y}_{entity}) \qquad \text{- 1 (Bunk, 2020)}$$

$L_{CRF}(.)$ gives us the negative log-likehood for a CRF

Intent classification

The outputted sequence received by the transformer for __CLS__ token as a_{CLS} and for the intent labels as y_{intent} are feed to the embedded layer which is a single semantic vector space h_{CLS} = E (a_{CLS}), h_{intent} = E (y_{intent}) where $h \epsilon$ IR (bunk, 2020).

For maximising the similarity S_I^+ we take the dot product loss with target label y_{intent}^+

$S_I^+ = h_{CLS}^T h_{intent}^+$

Similarly for minimising similarity S_I^- we take the dot product with negative samples y_{intent}^-

$S_I^- = h_{CLS}^T h_{intent}^-$

$$L_I = -\langle S_I^+ - \log\left(e^{S_I^+} + \sum_{\Omega_I^-} e^{S_I^-}\right)\rangle \qquad \text{- 2 (Bunk, 2020)}$$

From the above equation, we have taken the sum of negative samples Ω_I^- and for average we have consider all examples.

Masking

For predicting input tokens which is randomly masked an additional objective is introduced which was based on the masked language modelling task. 15% of all the input tokens are selected at random. Then the selected tokens are substituted with the mask token __MASK__ 70% of times, in 10% of cases the tokens are changed with random token and remaining are left original input. The output of transformer is like the output of intent loss and be given as (Bunk, 2020):

$$L_M = -\langle S_M^+ - \log\left(e^{S_M^+} + \sum_{\Omega_M^-} e^{S_M^-}\right)\rangle \qquad \text{- 3 (Bunk, 2020)}$$

where $S_M^+ = h_{MASK}^T h_{token}^+$ gives us the similarity with target label y_{token}^+, similarly for negative sample y_{token}^-. h_{MASK} = E (a_{MASK}) and h_{token} = E (a_{token}) are vector after been feed to the embedding layer. From the above, we have taken the sum of negative samples Ω_M^- and for average $\langle . \rangle$ we have considered all examples (Bunk, 2020).

Total loss

The training of our model is in multi-task fashion by decreasing the total loss L_{total}

$$L_{total} = L_I + L_E + L_M \qquad \text{- 4 (Bunk, 2020)}$$

Batching

Batching is done for handling the problem of class imbalances as it is observed that some of the intents have the grater tendency to occur compared to others. Additionally, we increase the batch size in overall training process as a part of the regularization (Bunk, 2020).

config.yml

```yaml
recipe: default.v1


language: en

pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 200
    constrain_similarities: true
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 200
    constrain_similarities: true
  - name: FallbackClassifier
    threshold: 0.3
    ambiguity_threshold: 0.1

policies:
  - name: MemoizationPolicy
  - name: RulePolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 200
    constrain_similarities: true
```

Figure 15. DIET Model Pipeline

The pipeline and policies are given in the config file as shown in figure 15 For converting into tokens and understanding the user input, we have used WhitespaceTokenizer, which tokenize on the base of the spacing between the words. Along with that we have used three featurizers. The RegexFeaturizer gives sparse features and token patterns from the input sequence. LexicalSynacticFeaturizer creates lexical and syntactic features for extracting entities out of it and CountVectorFeaturizer converts to Bag of Words representation. Additionally, EntitySynonymMapper is utilised for our model to know the synonyms in our data. ResponseSelector is responsible for our system to generate appropriate output with respect to user input.

Moreover, we have defined policies in our pipeline, each one contributes to our development of the chatbot. In general, through policies our conversational agent takes next action, these

policies can be based on the machine learning and rule based. The memorization policy predicts the following actions remembering the stories defined. Rule policy works in the event which are defined in the rules.yml file. For entity recognition and action prediction TED policy has been used.

3.2.6.2 Spacy Model

SpaCy (spacy.io, n.d.) is an open-source library used in problems related to Natural Language Processing (NLP). It is used in pre-processing and analyzing data in NLP. It offers several built-in components for named entity recognition, part of speech, dependency parsing, sentence segmentation, text classification, lemmatization, morphological analysis, entity linking etc. Additionally, it includes many features as follows:

- It can support up to 66 languages.
- State of the art speed and pretrained word vectors.
- Multi-task learning with pretrained transformers and linguistically motivated tokenization.
- Model packing and visualizers for syntax and NER.

For this RASA configurations we have used Spacy model “en_core_web_md”, where en is for the English language support, core can include (vocabulary, syntax, entities, vectors), web denoted the written text and md is for the size. Furtherly we have set the epoch to 200 for DIETClassfier, ResponseSelector. MemorizationPolicy for the chatbot to remember the stories, RulePolicy for execution for the defined rules and TEDPolicy for predicting the next action and entity recognition.

```yaml
language: "en"

pipeline:
  - name: SpacyNLP
    model: "en_core_web_md"
  - name: SpacyTokenizer
  - name: SpacyFeaturizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 200
    constrain_similarities: true
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 200
    constrain_similarities: true
  - name: FallbackClassifier
    threshold: 0.3
    ambiguity_threshold: 0.1

policies:
  - name: MemoizationPolicy
  - name: RulePolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 200
    constrain_similarities: true
```

Figure 16. SpacyNLP model pipeline

3.2.7 Rules Design

rules.yml

```yaml
version: "3.0"

rules:

- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye

- rule: Say 'I am a bot' anytime the user challenges
  steps:
  - intent: bot_challenge
  - action: utter_iamabot
```

Figure 17.Designing Rules

The figure 17 shows the following rules that has been defined in the rules.yml file. The first rule says if anytime the user responds with goodbye, then our system will respond with the same; our chatbot will predict the action utter_goodbye. Similarly, another rule will output with “I am a bot, powered by Rasa” on challenged by the user.

3.3 Integration

For making our chatbot accessible to banking customers we have integrated it on the WhatsApp. For integration we have used ngROK which acts as a server helping the chatbot to go online. And Twilio which provide a conversational API making conversation possible over WhatsApp. ngROK a cross-platform application that lets developer to host web server on the subdomain of ngrok.com, without any need of public IP or domain name.

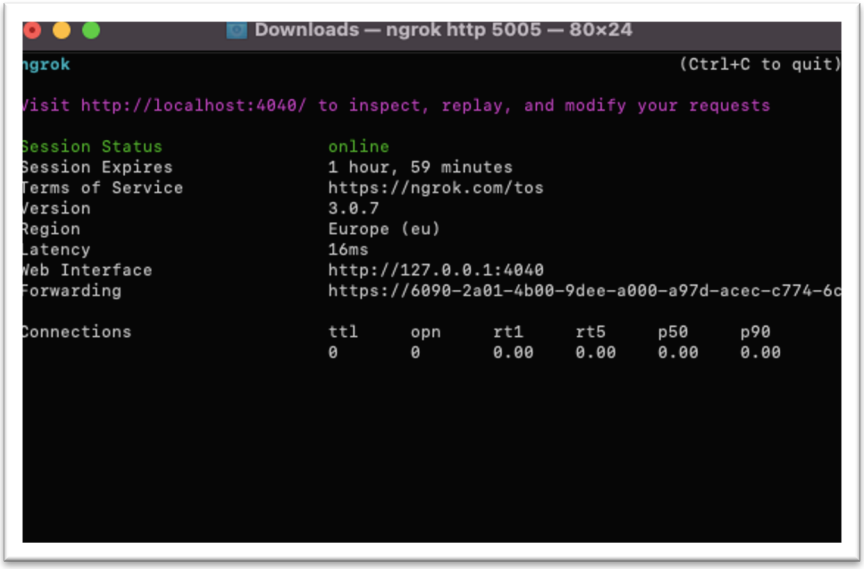

Figure 18. Starting a global server ngROK

Command to start a server on local machine

```
./ngrok http 5005
```

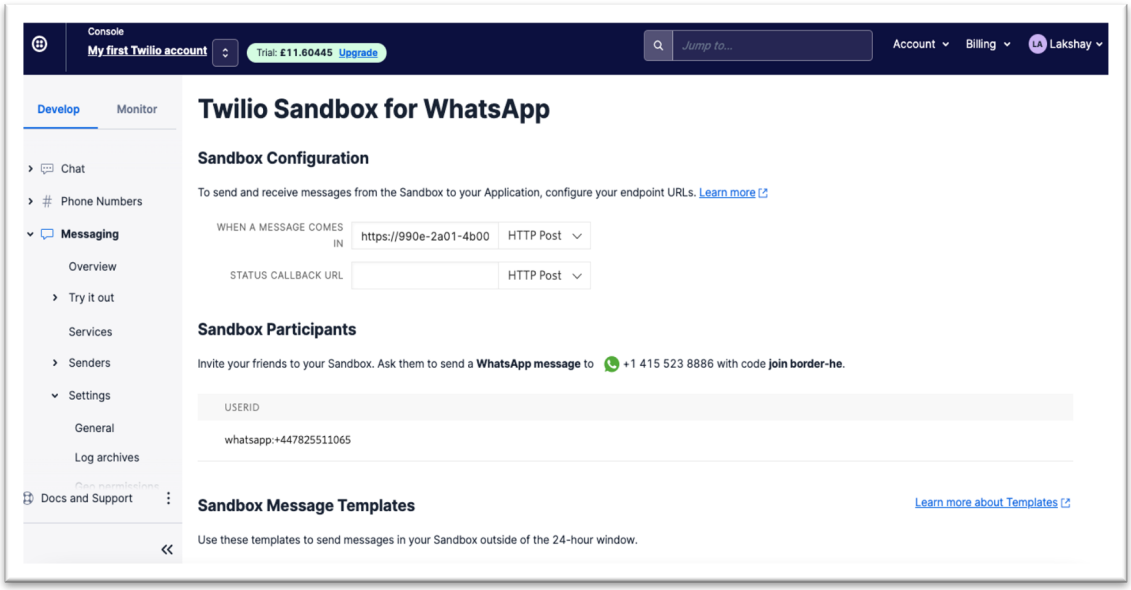

Figure 19. Twilio for WhatsApp integration

Additionally, Twilio is a communication platform as a service (CPaaS). It offers programmable Messaging API for WhatsApp. To use this, we have set up the sandbox configuration where we must provide the Hypertext Transfer Protocol Secure (https) from the ngROK server. Additionally, twilio require us configure the two-way messaging which lets user to send and receive messages. RASA framework support connector for twilio, which requires us to input the account SID, Auth Token and WhatsApp number on which the chatbot will send and receive requests.

Chapter 4: Evaluation

4.1 The Evaluation Metrix

Evaluation of our models is the important step in this dissertation. We have mainly focused on evaluating the NLU model and evaluating Dialogue model. For NLU model evaluation we have checked for intent recognition if our model is able to correctly identify the associated intent or not. And for Dialogue model evaluation, we have tested on a set of test stories for analysing the confidence of the models.

Classical metrics in the Machine Learning have been utilised for this work:

- **Precision:** It can be defined as the number of true positive instances among the total number of positive instances.
- **Recall:** It can be defined as number positive instances among the total number of instances.
- **F1-score:** It is a combination of precision and recall and can be calculated as below:

$$\mathrm{F1-score} = 2 \,.\, \frac{precision \,.\, recall}{precision + recall}$$

- **Confusion matrix:** It is a matrix-based table which displays the performance of a model. The row represents the actual class, and the column represents the predicted class.
- **Accuracy:** It can be found out by dividing the number of correct values over the total number of values.

The values of the above-mentioned matrices lie between 0 and 1 and values which are closer to 1 are better.

4.2 Evaluating NLU model

To evaluate the NLU model we have used the Held-Out Test Set, which is train- test set approach given in RASA framework. The following command given as (CI, 2022):

```
rasa data split nlu
```

This command divides the NLU dataset into train dataset and test dataset. The division is done on the bases that the test dataset should have 20% of NLU dataset and train dataset contains the rest 80% of NLU dataset.

Further to check the prediction by the NLU model on the test dataset we will run the command as (CI, 2022):

```
rasa test nlu
    --nlu train_test_split/test_data.yml
```

The result of this test is saved in Results and contains the following files:

- `intent_report.json`: It includes the report of every intent on its recall, f1-score, precision.
- `intent_confusion_matrix.png`: It gives us the intent confusion matrix which let us visualise the correctly and incorrectly predicted intents by our NLU model.
- `intent_histogram.png`: It gives us the intent histogram which displays confidence of each intent. The higher confidence is given by blue colour and lower confidence with red colour.
- `errors.json`: It lists the incorrectly predicted intents.

4.2.1 DIET Model Evaluation

The table gives result for Intent Evaluation-

Table 1. Intent Evaluation of DIET model

Intents	#train	#test	precision	recall	F1-score
Cash_machine	5	1	1.0	1.0	1.0
Credit_cards	6	1	1.0	1.0	1.0
Current_Accounts	14	1	1.0	1.0	1.0
Direct_debits	2	1	1.0	1.0	1.0
IBAN_BIC	5	1	1.0	1.0	1.0

International_Services	15	1	1.0	1.0	1.0
Loans	23	5	1.0	1.0	1.0
Mortgages	5	1	1.0	1.0	1.0
Overdrafts	9	1	1.0	1.0	1.0
Savings	15	1	1.0	1.0	1.0
Standing_orders	4	1	1.0	1.0	1.0
affirm	3	1	1.0	1.0	1.0
bot_challenge	3	1	1.0	1.0	1.0
change_contact_details	2	1	1.0	1.0	1.0
deny	6	1	1.0	1.0	1.0
difference_available_ledger_balance	4	1	1.0	1.0	1.0
dispute_transaction	5	1	1.0	1.0	1.0
first_name	10	1	1.0	1.0	1.0
goodbye	9	1	1.0	1.0	1.0
greet	12	1	1.0	1.0	1.0
last_name	9	1	1.0	1.0	1.0
money_limit	4	1	1.0	1.0	1.0
online_card_payment	5	1	0.5	1.0	0.66
options_Cards	1	1	1.0	1.0	1.0
options_Cards_AC	2	1	1.0	1.0	1.0
options_Cards_Pin	2	1	1.0	1.0	1.0
options_Credit_Cards	1	1	1.0	1.0	1.0
options_Declines	**1**	**1**	**0.0**	**0.0**	**0.0**
options_Different_query	1	1	1.0	1.0	1.0
options_Fee_charges	1	1	1.0	1.0	1.0
options_Pay_transfer	2	1	1.0	1.0	1.0
options_Payments	1	1	1.0	1.0	1.0
options_Personal_loan	1	1	1.0	1.0	1.0
options_Product_Lending	2	1	1.0	1.0	1.0
options_account	1	1	1.0	1.0	1.0
options_charges	1	1	1.0	1.0	1.0
options_cheques	1	1	1.0	1.0	1.0
options_disputes	1	1	1.0	1.0	1.0
options_disputes_ED	1	1	1.0	1.0	1.0
options_disputes_ND	1	1	1.0	1.0	1.0
options_fees	1	1	1.0	1.0	1.0
options_savings	1	1	1.0	1.0	1.0
options_statements	1	1	1.0	1.0	1.0
pending_and_upcoming_transactions	4	1	1.0	1.0	1.0
storedatabase	9	1	1.0	1.0	1.0

thanks	2	1	1.0	1.0	1.0
	166	50	96.7	97.8	97
Accuracy = 97.8					

Intent Confusion Matrix and Intent Prediction Confidence Distribution

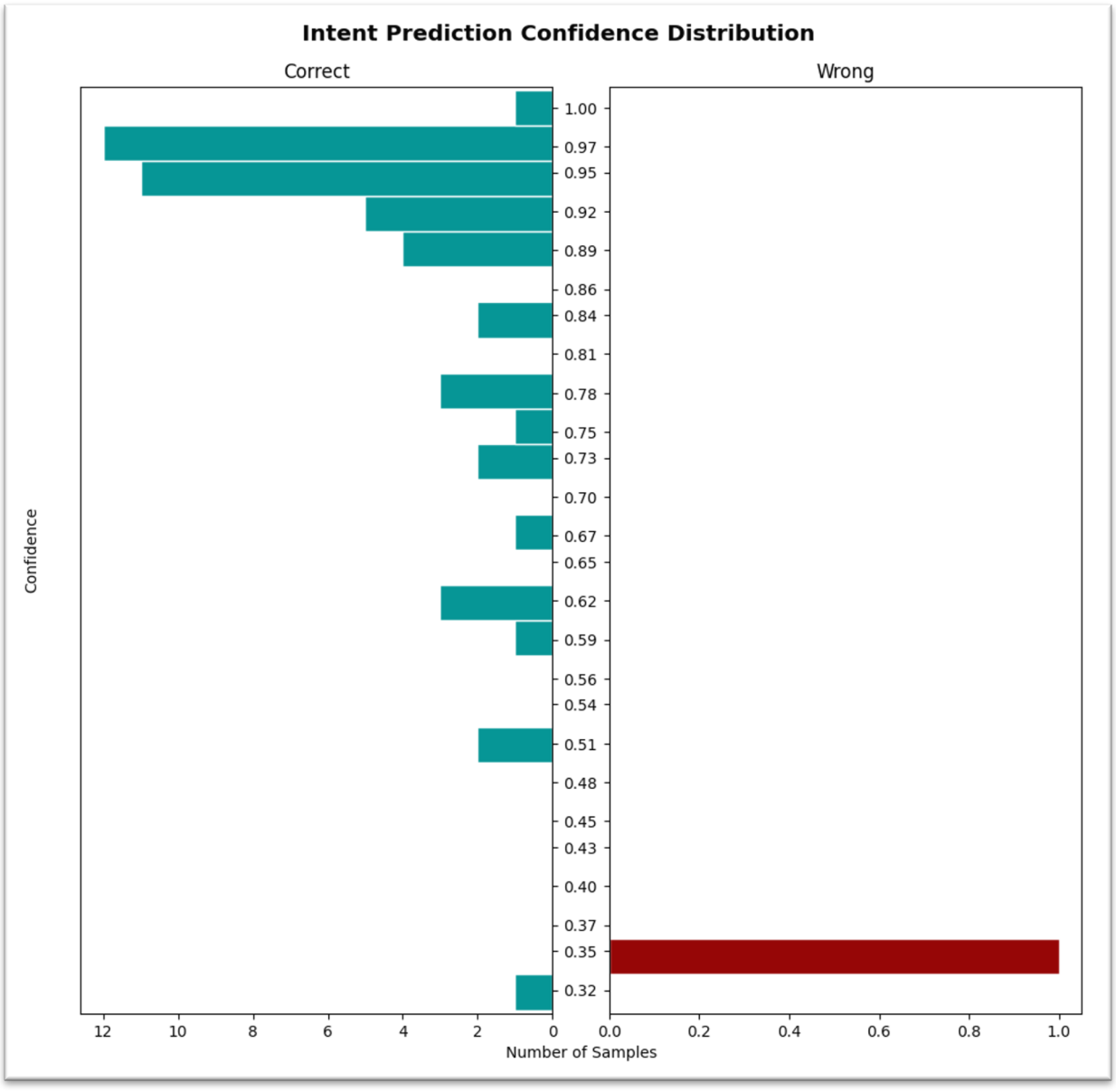

Figure 20. Intent prediction confidence distribution for DIET model

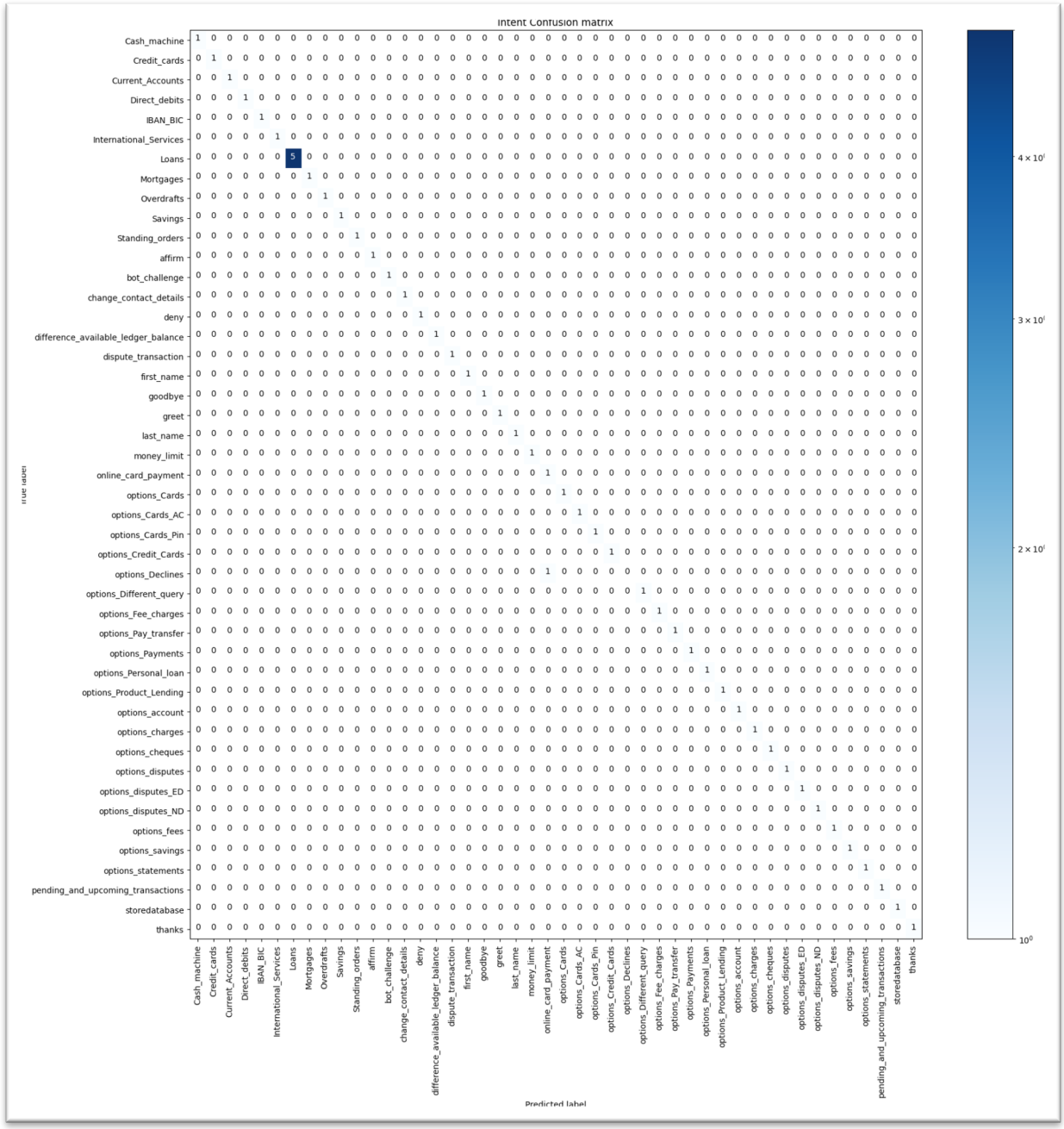

Figure 21. Intent confusion matrix for DIET model

4.2.2 Spacy Model Evaluation

The table gives result for Intent Evaluation-

Table 2. Intent Evaluation of SpacyNLP model

Intents	#train	#test	precision	recall	F1-score
Cash_machine	5	1	1.0	1.0	1.0
Credit_cards	6	1	0.33	1.0	0.5
Current_Accounts	14	1	1.0	1.0	1.0
Direct_debits	2	1	1.0	1.0	1.0
IBAN_BIC	5	1	1.0	1.0	1.0
International_Services	15	1	1.0	1.0	1.0
Loans	23	5	0.83	1.0	0.90
Mortgages	5	1	1.0	1.0	1.0
Overdrafts	9	1	1.0	1.0	1.0
Savings	15	1	1.0	1.0	1.0
Standing_orders	4	1	1.0	1.0	1.0
affirm	3	1	1.0	1.0	1.0
bot_challenge	3	1	1.0	1.0	1.0
change_contact_details	2	1	1.0	1.0	1.0
deny	6	1	1.0	1.0	1.0
difference_available_ledger_balance	4	1	1.0	1.0	1.0
dispute_transaction	5	1	1.0	1.0	1.0
first_name	10	1	1.0	1.0	1.0
goodbye	9	1	1.0	1.0	1.0
greet	12	1	1.0	1.0	1.0
last_name	9	1	1.0	1.0	1.0
money_limit	4	1	1.0	1.0	1.0
online_card_payment	5	1	0.5	1.0	0.66
options_Cards	**1**	**1**	**0.0**	**0.0**	**0.0**
options_Cards_AC	2	1	1.0	1.0	1.0
options_Cards_Pin	2	1	1.0	1.0	1.0
options_Credit_Cards	**1**	**1**	**0.0**	**0.0**	**0.0**
options_Declines	1	1	1.0	1.0	1.0
options_Different_query	1	1	1.0	1.0	1.0
options_Fee_charges	1	1	1.0	1.0	1.0
options_Pay_transfer	2	1	1.0	1.0	1.0
options_Payments	1	1	1.0	1.0	1.0

options_Personal_loan	**1**	**1**	**0.0**	**0.0**	**0.0**
options_Product_Lending	2	1	1.0	1.0	1.0
options_account	1	1	1.0	1.0	1.0
options_charges	1	1	1.0	1.0	1.0
options_cheques	1	1	1.0	1.0	1.0
options_disputes	1	1	1.0	1.0	1.0
options_disputes_ED	**1**	**1**	**0.0**	**0.0**	**0.0**
options_disputes_ND	1	1	0.5	1.0	0.66
options_fees	1	1	1.0	1.0	1.0
options_savings	1	1	1.0	1.0	1.0
options_statements	1	1	1.0	1.0	1.0
pending_and_upcoming_transactions	4	1	1.0	1.0	1.0
storedatabase	9	1	1.0	1.0	1.0
thanks	2	1	1.0	1.0	1.0
	166	50			
			87.3	91.3	88.5
Accuracy = 91.3					

Intent Confusion Matrix and Intent Prediction Confidence Distribution

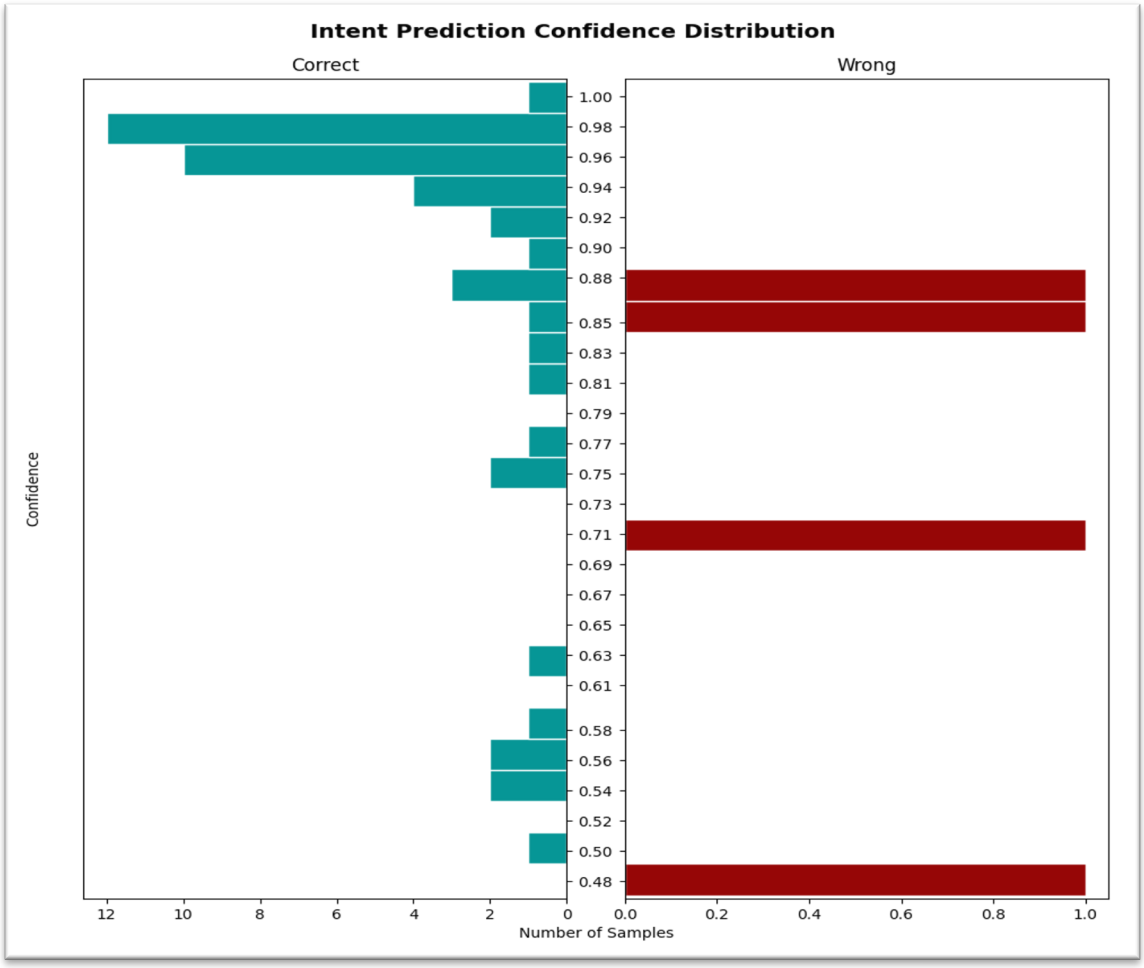

Figure 22. Intent prediction confidence distribution for SpacyNLP model

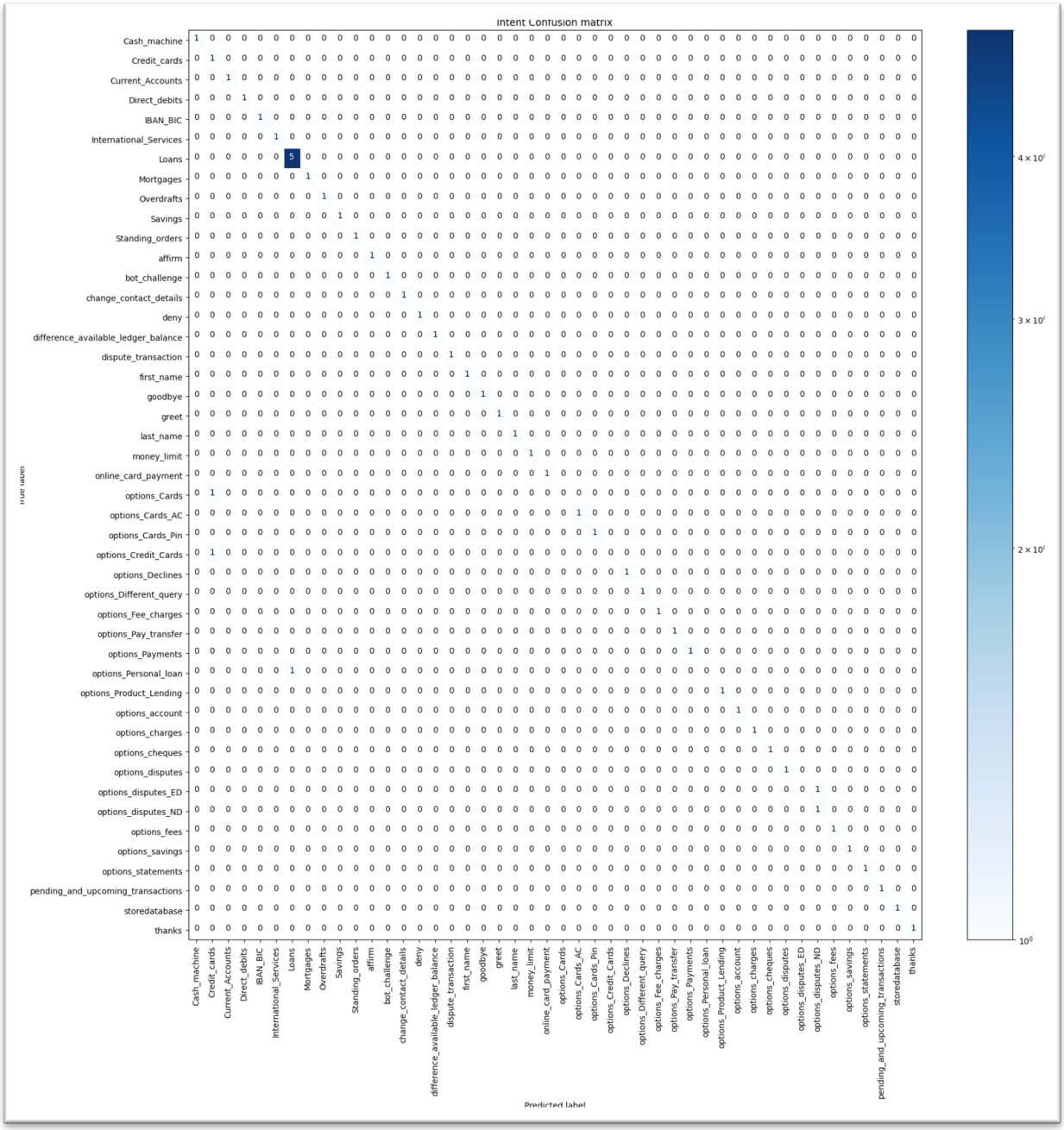

Figure 23. Intent confusion matrix for SpacyNLP model

4.2.3 Results

From the results we can conclude that the both the DIET model and SpacyNLP model exceeded the accuracy of more than 90%. Although overall the DIET model performed the best in our case and was able to give **precision (96.7), recall (97.8) and F1-score (97)** compared to results of other model that gave **precision (87.3), recall (91.3) and F1-score (88.5).**

From the intent prediction confidence distribution of both the models we can see that most the intents lie in the high confidence space but in DIET configuration there only 2 intents which lie in 51-0 range. Additionally, there is 1 intent (options_Declines) which is wrongly predicted compared to 4 intents (options_Cards, options_Credit_Cards, options_Personal_loan, options_disputes_ED) by SpacyNLP model.

The possible reason for the SpaCy model to not outperform, as we selected model "en_core_web_md" which is pretrained on textual data that combines blogs, comments and news considering to our domain which banking.

4.3 Evaluating Dialogue Model

Next, we will evaluate Dialogue model on a set of test stories for both our models. Additionally, if while testing some stories fail, they will automatically be saved in results file. The report will include failed stories along with the confusion matrix for each story (CI, 2022).

For command for testing test stories as follows (CI, 2022):

```
rasa test core --stories test_stories.yml --out results
```

The result of this test is saved in Results and contains the following files:

- `failed_test_stories.yml`: The file consists of all the incorrectly predicted stories.
- `story_confusion_matrix.png`: The file consists of action confusion matrix.
- `story_report.json`: It includes the report of every action on its recall, f1-score, precision.

4.3.1 DIET Model Evaluation

Action Confusion Matrix

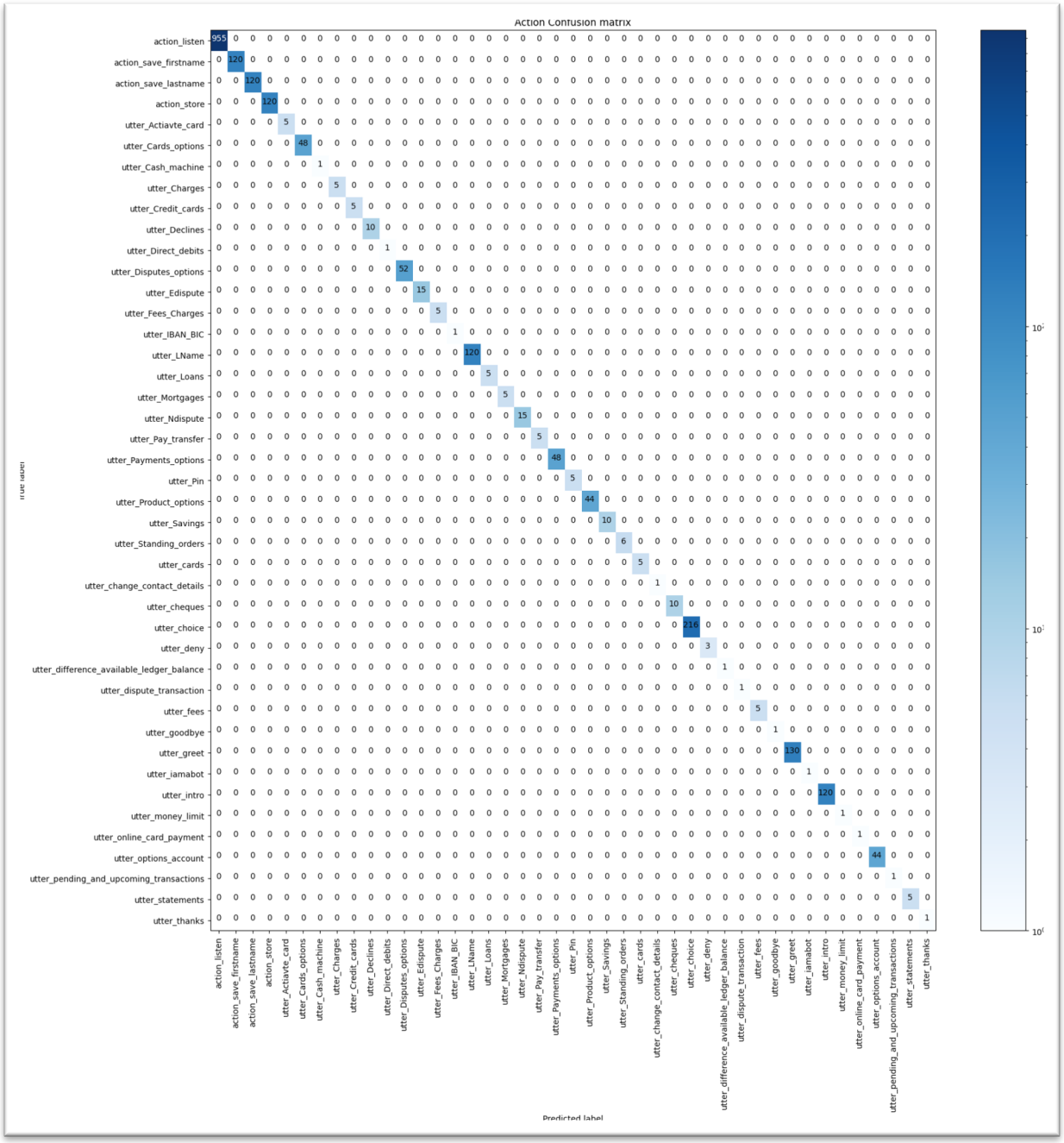

Figure 24. Action confusion matrix for DIET model

4.3.2 Spacy Model Evaluation

Action Confusion Matrix

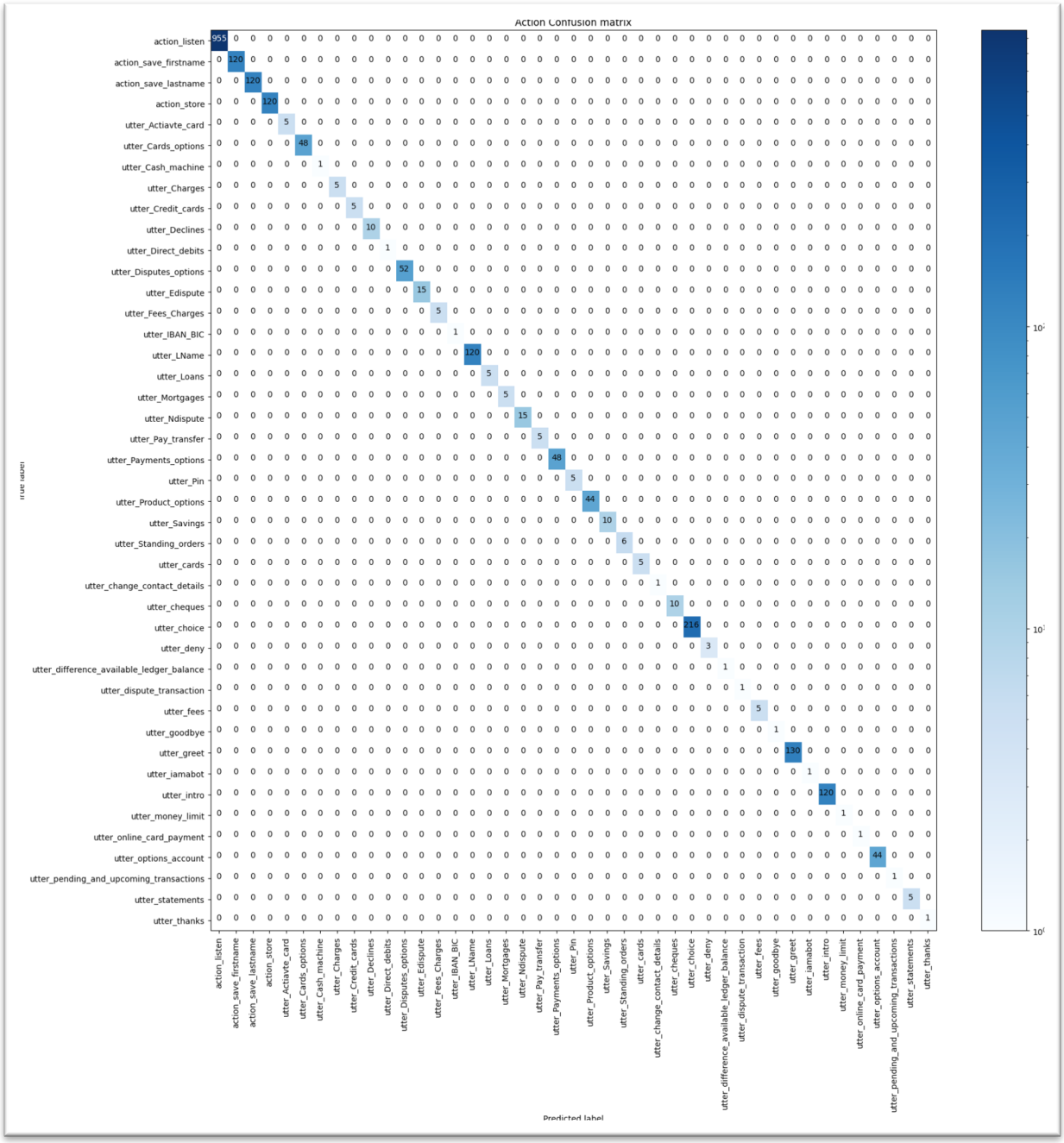

Figure 25. Action confusion matrix for SpacyNLP model

4.3.3 Results

The table below represents the overview of results for intent prediction.

Table 3. Results

Models	#train	#test	precision	recall	F1-score
DIET	**166**	**50**	**96.7**	**97.8**	**97**
SpacyNLP	166	50	87.3	91.3	88.5

From the results from the **stories evaluation**, we can conclude that both the models performed the same in predicting stories with no failed stories. But as the default setting gave us the best results overall therefore, we will use the **DIET model** in our development of the chatbot.

Chapter 5: Results

This chapter will include the final chatbot prototype that we have used in the development of banking financial customer support chatbot. It will provide an overview of the product in real world setting.

5.1 Final Prototype

This research aimed to build a banking chatbot which could handle all the queries related to banking in no time. For the same we have included a global server ngROK which is responsible for our chatbot to go online. Additionally, for providing the WhatsApp API we have used Twilio is lets us connect our chatbot with WhatsApp business. Below are some pictures displaying the working of our chatbot in the real-world setting, which our targeted customer would experience.

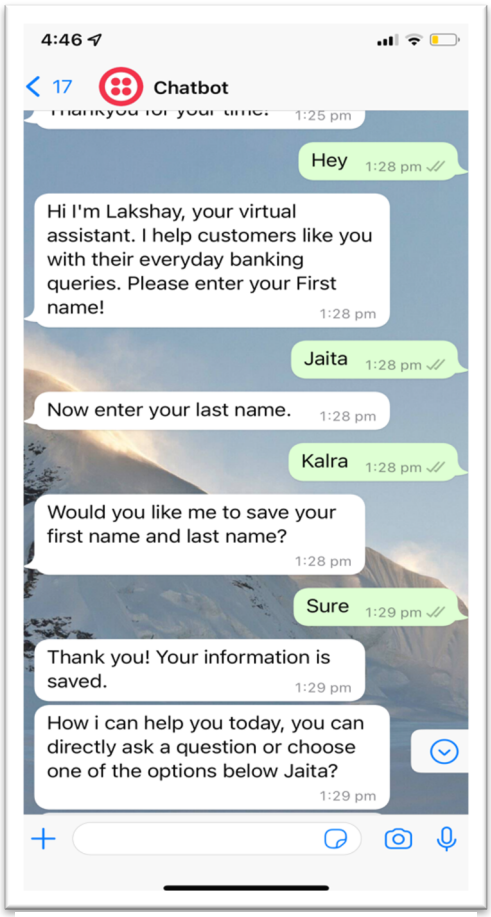

Figure 26. Response "Saving user's information"

The above picture displays the conversational flow of our chatbot. For the start the user starts the conversation with “Hey”, moving forward the chatbot lets the user to input their details namely firstname and lastname and saves the provided information into the database (sqllite 3). Further we have defined the stories in such a way that it lets user to query question following two paths.

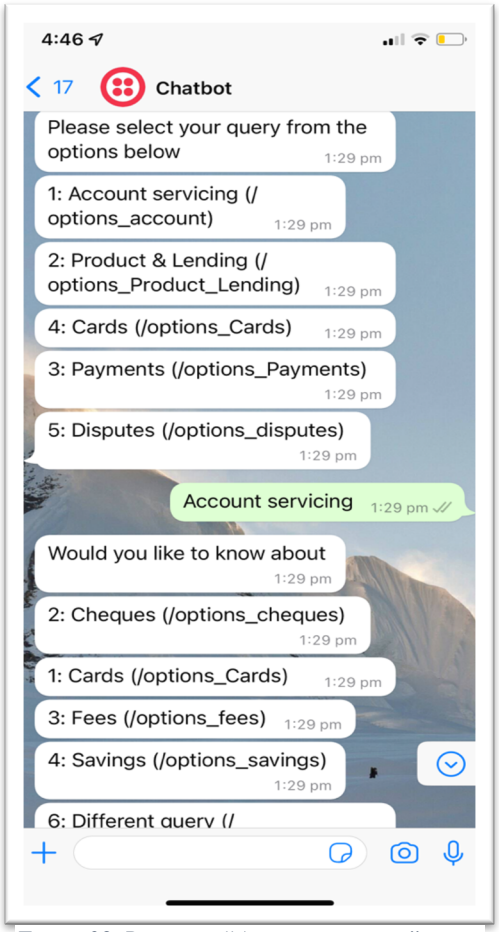

Figure 28. Response “Account servicing”

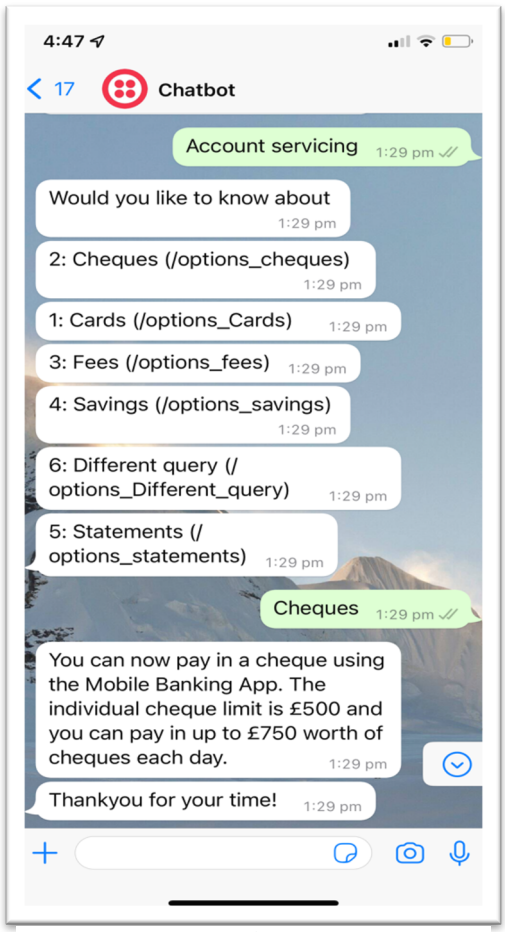

Figure 27. Response “Cheques”

The above pictures show us the first path which the user can choose. Here the user wants seeks a solution for the query which is related to cheques. As it is a bottom based chatbot, the user pops up with several options on every choice he/she makes. The user selects an option in which the respective query falls in, for example here the user selected “Account serving” which further displays options to choose from: Cards, Cheques, Fees, Savings, Statements and Different query. As the user wanted to know about cheques, it will be selected for the results.

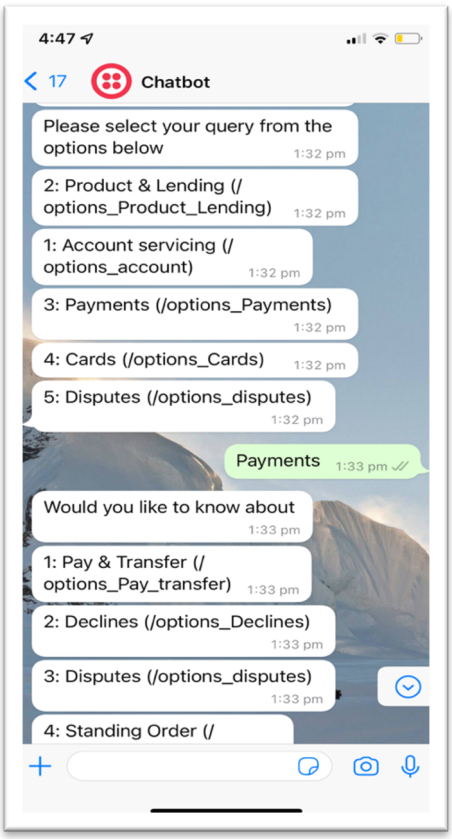

Figure 29. Response "Payments"

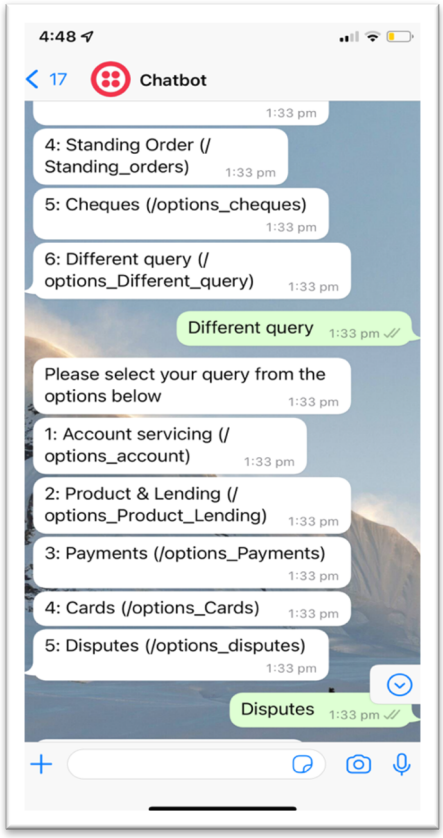

Figure 30. Response "Different query"

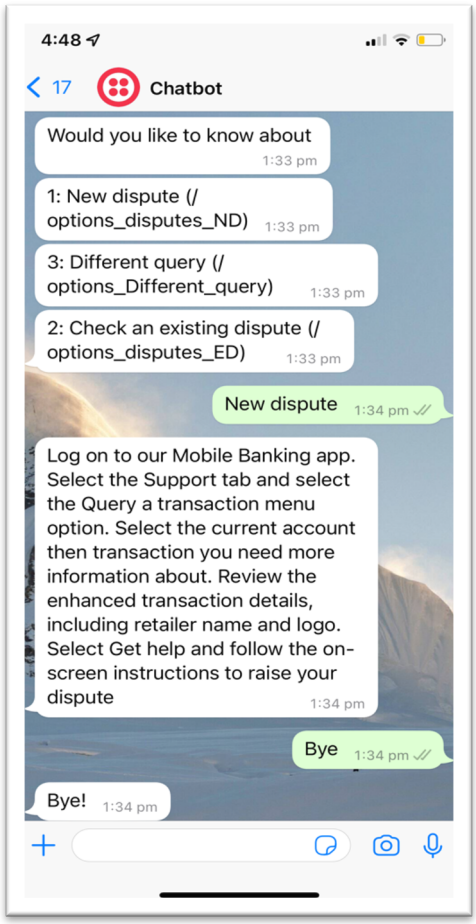

Figure 31. Response "New dispute"

The example of second path which the user can take is displayed in the above three pictures. After being into main window where the user selects “Payments” option but then decided to ask a different query for this case, we have included an option of “Different query” which lets user to back propagate to the main menu, from there user can take the next indented path for querying out the results.

Chapter 6: Conclusion and Future Work

This chapter will provide an overview with all the work done in the previous sections. It will be focused on the discussing the conclusions and the future development in this work.

6.1 Conclusion

The project aimed to develop a banking chatbot that could handle the incoming queries of the customer in the minimum expected time. For coping up with our objectives we have discovered the required research and background in the development of the chatbots. From literature review we have observed the gaps in the banking industry. Generative models and the techniques being utilised in the development of the chatbot. The exploration of RASA framework which uses concepts of transfer learning and fine turning for the development industrial based chatbots.

In methodology we have discussed the design and development of chatbot. To achieve the objective of providing constant support to the customers we have integrated our chatbot to WhatsApp. The integration of Chatbot is done using the gobal server ngROK and Twilio which provide the API for WhatsApp messaging. Additionally, all the banking queries are handled by our chatbot without the indulgence of human interactions let us achieve the objective of increasing the efficiency of banking personnel.

Consistent answers are given by our chatbot every time on the user request making use of rule-based systems. Moreover, we have combined the flexibility of bottoms which lets user to select from the given number of options, this offers advantages like; the user has less changes to diverge in the conversation and this let users to get results a lot quicker compared to traditional systems. Chatbot being available on the fingertips lets the customer to seek the required information without getting into a hassle of physical present on the premises.

Further we performed evaluation of the chatbot major on two bases, first on the intent prediction and second on the conversation evaluation (stories evaluation). For the intent evaluating we have performed unit testing considering the DIET model and SpacyNLP model where the DIET model pipeline gave us the best results in intent prediction.

Additionally, the Dialogue model is evaluated comparing the prediction of stories from both the pipelines, where both the pipelines gave us similar results.

6.2 Future Work

Chatbots hold a lot of potential in the future, they will become more human like. Self-learning, predicting the human behaviour more accurately and reliability in terms of securing information are some of the advancements will be discussed further in this section.

- For targeting the wider audience, we can provide the support for several languages by using state of the art algorithms which are pre-trained on multiple languages.
- Chatbots could be more reliable, handling the information securely without compromising on the personal details of customers.
- Various features can be included that would provide the flexibility for users to query about complex tasks such as investing, opening an FD, request for extending credit card limit etc.
- Moreover, additional data can be taken into consideration for the development of the chatbot.

References

1. Fauzia, L. and Hadiprakoso, R.B., 2021, December. Implementation of Chatbot on University Website Using RASA Framework. In *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)* (pp. 373-378). IEEE.

2. Kulkarni, C.S., Bhavsar, A.U., Pingale, S.R. and Kumbhar, S.S., 2017. BANK CHAT BOT–an intelligent assistant system using NLP and machine learning. *International Research Journal of Engineering and Technology*, *4*(5), pp.2374-2377.

3. Hwang, S. and Kim, J. (2021) Toward a Chatbot for Financial Sustainability, *Sustainability*, MDPI AG, 13(6), p. 3173, [online] Available at: http://dx.doi.org/10.3390/su13063173.

4. Singh, M., Singh, R., Pandey, A. and Kasture, P., 2018. Chat-Bot for Banking Industry. In *International Conference on Communication, Security and Optimization of Decision Support Systems (ICCSOD 2018)* (pp. 247-249).

5. Meshram, S., Naik, N., Megha, V.R., More, T. and Kharche, S., 2021, August. College Enquiry Chatbot using Rasa Framework. In *2021 Asian Conference on Innovation in Technology (ASIANCON)* (pp. 1-8). IEEE.

6. Sharma, R.K. and Joshi, M., 2020. An analytical study and review of open source chatbot framework, RASA. *International Journal of Engineering Research and*, *9*(06).

7. Lester, J., Branting, K. and Mott, B., 2004. Conversational agents. *The practical handbook of internet computing*, pp.220-240.

8. Gupta, A., Hathwar, D. and Vijayakumar, A., 2020. Introduction to AI chatbots. *International Journal of Engineering Research and Technology*, *9*(7), pp.255-258.

9. Introduction to Rasa Open Source, (2022) *Rasa.com*, [online] Available at: https://rasa.com/docs/rasa/ (Accessed 29 August 2022).

10. Mauldin, M.L., 1994, August. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In *AAAI*(Vol. 94, pp. 16-21).

11. Leah (2022). *How Rule-Based Chatbots Beat AI Chatbots in Business*. [online] Userlike Live Chat. Available at: https://www.userlike.com/en/blog/rule-based-chatbots#versus [Accessed 19 Aug. 2022].

12. Sharma, Vibhor, et al. "An Intelligent Behaviour Shown by Chatbot System." International Journal of New Technology and Research, vol. 3, no. 4, Apr. 2017

13. Mandasari, Y., “Follow-up Question Generation”. MA thesis. University of Twente, 2019

14. Wallace, R., 2003. The elements of AIML style. *Alice AI Foundation*, *139*.

15. Bapat, R., "Helping Chatbots To Better Understand User Requests Efficiently Using Human Computation". MSc. thesis. Delft University of Technology, 2017

16. Traum, D.R., Larsson, S. (2003). The Information State Approach to Dialogue Management. In: van Kuppevelt, J., Smith, R.W. (eds) Current and New Directions in Discourse and Dialogue. Text, Speech and Language Technology, vol 22. Springer, Dordrecht. https://doi.org/10.1007/978-94-010-0019-2_15

17. Telang, P.R., Kalia, A.K., Vukovic, M., Pandita, R. and Singh, M.P. (2018). A Conceptual Framework for Engineering Chatbots. *IEEE Internet Computing*, 22(6), pp.54–59. doi:10.1109/mic.2018.2877827.

18. QUARTERONI, S. and MANANDHAR, S. (2009) "Designing an interactive open-domain question answering system," *Natural Language Engineering*. Cambridge University Press, 15(1), pp. 73–95. doi: 10.1017/S1351324908004919.

19. Goddeau, D., Meng, H., Polifroni, J., Seneff, S. and Busayapongchai, S., 1996, October. A form-based dialogue manager for spoken language applications. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96* (Vol. 2, pp. 701-704). IEEE.

20. Griol, D., Callejas, Z., López-Cózar, R. and Riccardi, G., 2014. A domain-independent statistical methodology for dialog management in spoken dialog systems. *Computer Speech & Language*, *28*(3), pp.743-768.

21. Adamopoulou, E. and Moussiades, L., 2020, June. An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 373-383). Springer, Cham.

22. Roodschild, M., Gotay Sardiñas, J. and Will, A., 2020. A new approach for the vanishing gradient problem on sigmoid activation. *Progress in Artificial Intelligence*, *9*(4), pp.351-360.

23. Sutskever, I., Martens, J. and Hinton, G.E., 2011, January. Generating text with recurrent neural networks. In *ICML*.

24. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

25. Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

26. Rahutomo, F., Kitasuka, T. and Aritsugi, M., 2012, October. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST* (Vol. 4, No. 1, p. 1).

27. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. and He, Q., 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), pp.43-76.

28. Church, K.W., Chen, Z. and Ma, Y., 2021. Emerging trends: A gentle introduction to fine-tuning. *Natural Language Engineering*, *27*(6), pp.763-778.

29. Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

30. Prakash, A., Hasan, S.A., Lee, K., Datla, V., Qadir, A., Liu, J. and Farri, O., 2016. Neural paraphrase generation with stacked residual LSTM networks. *arXiv preprint arXiv:1610.03098*.

31. Mathur, V. and Singh, A., 2018. The rapidly changing landscape of conversational agents. *arXiv preprint arXiv:1803.08419*.

32. Zhang, X. and Wang, H., 2016, July. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI* (Vol. 16, No. 2016, pp. 2993-2999).

33. Chen, Q., Zhuo, Z. and Wang, W., 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

34. Niu, P., Chen, Z. and Song, M., 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. *arXiv preprint arXiv:1907.00390*.

35. Vanzo, A., Bastianelli, E. and Lemon, O., 2019. Hierarchical multi-task natural language understanding for cross-domain conversational ai: HERMIT NLU. *arXiv preprint arXiv:1910.00912*.

36. Baker, C.F., Fillmore, C.J. and Lowe, J.B., 1998. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

37. Bunk, T., Varshneya, D., Vlasov, V. and Nichol, A., 2020. Diet: Lightweight language understanding for dialogue systems. *arXiv preprint arXiv:2004.09936*.

38. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, *30*.

39. Components, (2022) *Rasa.com*, [online] Available at: https://rasa.com/docs/rasa/components/ (Accessed 29 August 2022).

40. Luo, B., Lau, R.Y., Li, C. and Si, Y.W., 2022. A critical review of state-of-the-art chatbot designs and applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *12*(1), p.e1434.

41. Hien, H.T., Cuong, P.N., Nam, L.N.H., Nhung, H.L.T.K. and Thang, L.D., 2018, December. Intelligent assistants in higher-education environments: the FIT-EBot, a chatbot for administrative and learning support. In *Proceedings of the ninth international symposium on information and communication technology* (pp. 69-76).

42. Adamopoulou, E. and Moussiades, L., 2020, June. An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 373-383). Springer, Cham.

43. Smarr, J.S., Grenager, T.G., Levy, R. and Manning, C. (n.d.). *WhitespaceTokenizer (Stanford JavaNLP API)*. [online] nlp.stanford.edu. Available at: https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/process/WhitespaceTokenizer.html [Accessed 1 Sep. 2022].

44. Policies, (2022) *Rasa.com*, [online] Available at: https://rasa.com/docs/rasa/policies (Accessed 29 August 2022).

45. www.hsbc.co.uk. (n.d.). *HSBC UK - Personal & Online Banking*. [online] Available at: https://www.hsbc.co.uk [Accessed 3 Sep. 2022].

46. Actions, (2022) *Rasa.com*, [online] Available at: https://rasa.com/docs/rasa/actions (Accessed 29 August 2022).

47. spacy.io. (n.d.). *spaCy · Industrial-strength Natural Language Processing in Python*. [online] Available at: https://spacy.io [Accessed 8 Sep. 2022].

48. CI, G. (2022). *Testing Your Assistant*. [online] rasa.com. Available at: https://rasa.com/docs/rasa/testing-your-assistant#evaluating-an-nlu-model [Accessed 9 Sep. 2022].