ACT CHATBOT IMPROVEMENT USING DATA ANALYSIS

A Project Report

submitted by

J OM SHRI PRASATH

in partial fulfilment of the requirements
for the award of the degree of

of

BACHELOR OF TECHNOLOGY
&
MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING & DATA SCIENCE
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2022

THESIS CERTIFICATE

This is to certify that the thesis titled **ACT CHATBOT IMPROVEMENT USING DATA ANALYSIS**, submitted by **J OM SHRI PRASATH**, to the Indian Institute of Technology, Madras, for the award of the degree of **B.Tech** and **M.Tech**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Nandan Sudarsanam
Project Guide
Professor
Dept. of Management Studies
IIT Madras, 600 036

Place: Chennai

Date: 17 June 2022

ACKNOWLEDGEMENTS

I would like to extend my sincere thanks to my guide - Prof. Nandan Sudarsanam, for his constant guidance throughout the project. His valuable support in the course of the project and his unique insights in analysing various aspects of the problem statement helped immensely in my work and also widened my horizon to different aspects of data science. Without his contributions, the project would have not come to the current state it has reached. I would also like to thank my parents and my brother for their constant support throughout the uncertain times of COVID.

ABSTRACT

KEYWORDS: Chatbot, Analysis, Sequence Mining, HMM, NLP

The project aim to improve the current existing rule-based chatbot of ACT based on data analysis. First, analysis of the user interaction with the chabot was done by pre-processing the data into different building blocks of sequential data with the help of sequence mining. Later the sequences were modelled as HMM and analysed based on their transitions. The user responses to internet issues were analysed using NLP and based on the analysis, suggestions to improve the chabot were found.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACT	**A**tria **C**onvergence **T**echnologies
PrefixSpan	(**Prefix**-projected **S**equential **pa**ttern mining)
HMM	**H**idden **M**arkov **M**odel
JSON	**J**ava**S**cript **O**bject **N**otation
TF-IDF	**T**ext **F**requency - **I**nverse **D**ocument **F**requency

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

ACT operates a rule based chatbot for consumer queries on their website. Based on the data collected from interactions between the chatbot and the user, improvements to the chatbot to enhance user experience by reducing the amount of time spent in interacting with the chatbot are to be suggested.

1.2 Chatbots

Chatbots are computer programs designed to hold a online conversation with humans through a text-based interface. They are usually an dialog system which are used for various purposes like customer service, information retrieval, personal assistance and more recently as a tool for customer interactions by major e-commerce retailers. The major advantages of chatbot are their 24/7 availability which means that consumers can be attended to round the clock, they provide easy automation for users to do their tasks for them like setting alarm, order tickets for movies and so on, they are cost-effective as they can handle the work of many humans at once thus saving cost and they can be used to gain useful information from the consumers which can help businesses to cater to their customers better. The major disadvantage of chatbots are that they are robotic and cannot be programmed to handle all possible consumer queries on their own and might need to divert the users to humans to handle the situation better. This disadvantage is slowly being handled by advances in machine learning and natural language processing.

There are two types of chatbots based on their implementation which are :

- **Rule Based Chatbot :** Chatbots which operate based on fixed rules and are usually operated by giving users options to select from or severely restrict the inputs which can be accepted from the user. These chatbots have low chances of giving wrong response to users as their inputs and outputs are predetermined.
- **Self Learning Chatbot :** Chatbots which use machine learning algorithms to understand the user query and respond to it. These chatbots can take in any type of natural language input from users, but can many times go wrong in their output

which can be unrelated to the user query.

Due to the volatile nature of the self-learning chatbots, rule based chatbots are the ones most used in the industry, although recent advances in machine learning has caused quite a shift from rule based chatbot to self learning chatbots. ACT uses an rule-based chatbot, thus we can look more into it.

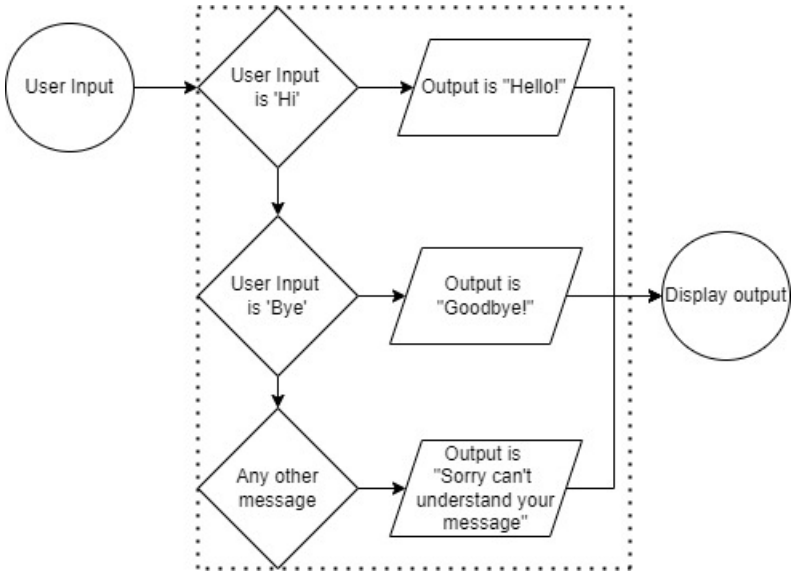

Figure 1.1: Rule Based Chatbot Example

Rule based chatbots usually built using a framework which works based on a if-else based logic. The user is allowed to either enter a fixed set of natural language input which are close enough to what can be accepted as input by the system or they are provided with predefined set of options to choose from by the chatbot. Although restrictive, this makes sure that there is little to no room for error by the chatbot in responding to a user query. Many rule based chatbots also use machine learning in case they need to understand the user query better in case the input is natural language. ACT operates based on predefined input type chatbot which provides a fixed set of inputs at each stage of the conversation for the user.

1.3 Literature Review

Literature on improving chatbots based on data analysis are scarce. Many of these literature on rule based chatbots have been on improving the chatbot response to users by using natural language processing and machine learning techniques. Følstad and Taylor (2021) creates metrics for qualitative analysis of chatbots based on response relevance and outcome and also analyse the chatbot messages in a sequential way. We will use some of these techniques modified for our use case to analyse the dataset. Efraim *et al.* (2017) try to improve a answer retrieval chatbot using user satisfaction as the metric, but their work revolves around improving the natural language aspect of the chatbot since users type in queries in their model of chatbot which is different from our case where options are provided for users to choose from. But a few of the natural language aspect of the work can be used in analysing user responses to internet issues in Section 3.2. Peng and Ma (2019) cites different methods of improving chatbots using human chatbot collaboration and making chatbots learn from users, but these methods are for general chatbots and do not apply to our specific case where business level suggestions for user experience enhancement is required.

As discussed many of the chatbots are working on the principles of improving user experience based on response improvement of the chatbot. Since in our case the responses of the chatbot are fixed already in the framework, the better way to improve the user experience in this case would be to analyse the sequences of interactions between the users and chatbot to come up with suggestions which can help reduce the number of times the users need to access the chatbot to solve their issues. This can be achieved by analysing user behaviour while interacting with the chatbot based on the options they choose the most and see if there are any places in the options where the users can be provided with more information to reduce their interactions with the chatbot or some improvements in the flow design which can help the users communicate more effectively with the chatbot.

1.4 Dataset

The data provided by ACT was of JSON format where each entry denoted a single point of back-and-forth between the user and the chatbot. The fields are as follows:

Field	**Details**
_id	ID of the interaction in the table
api_key	Empty field
mobile	Empty field
authorized	Had only 1 as value
headers	Header metadata involved in the interaction
id	ID of the interaction in the table
ip_address	IP Address associated with interaction
method	Had only POST as value
params	Parameters used for POST
response_code	Response code of POST method
response_data	Response data of the interaction
rtime	Response Time
server_ip	IP Address of server
time	Time of interaction
uri	URI for the POST method

Table 1.1: Fields present in chatbot data

This dataset is not in the format of chat sessions, but instead contains a mix of details regarding single interaction between an user and a chatbot. Thus, converting the dataset into chat-like sequence is necessary for further analysis. Also, another thing to note is that there are no natural language conversations between the user and chatbot, only selection of options. Thus to represent the chat sessions we need to condense the data into a proper representation of a chat based on the option selected by the user. On analysing the data, **response_data**, **headers**, **time**, **params** and **uri** are the fields which are useful for extracting and processing the data, with **response_code** used to remove failed interactions due to network error. For further analysis, we use data from 1st to 3rd March and Python with its various packages will also be used.

CHAPTER 2

DATA PREPROCESSING

The above raw data needs to be preprocessed for further analysis. We need to define few constituents which are the result of preprocessing the above data, and also look at how to extract them from the raw data and how they are connected with each other.

2.1 Data Constituents

Interaction

Interaction denotes a particular action between the user and the chatbot. This is represented by a single entry in the above data. Each interaction involves the chatbot giving user different options or some information and the user responding to proceed further. An example would be the chatbot displaying the remaining data balance of the user, or asking user what is the type of internet issue they are facing.

Engagement

Engagement denotes a string of interactions by a particular user starting from user initiating contact with the chatbot till user stops using the chatbot. For example, an engagement involves the following interactions : user logging in, checking internet data balance used and requesting to send bill to their email.

Multi-Engagement

Multi-Engagement denotes when the user tries to attempt multiple engagements with the chatbot in quick successions. These engagements can be joined together to denote a new type of engagement called

Objective

Objective denotes the intent behind certain string of interactions. These strings of interactions have been decided from commonly occurring strings of interactions which are prevalent in the engagement.

2.2 Representing Interactions

Each datapoint in the above dataset constitutes a interaction between user and chatbot. But for further analysis purpose, we need to One thing to note is that the data presented contains the output shown by the chatbot to the user and and what the user replied with, but most of the time the user response is related to what the next interaction consists of so for analysis purposes, only using the chatbot output to represent what the interaction consists of is enough to understand the flow of the chat. To represent the interaction in a more amenable way, we represent the interaction as a short string which describes what the interaction was about in as short a way as possible. For example, assume in an interaction the chatbot shows the user how much data balance has been used till now, this interaction is represented as *DataUsed*. If the chatbot tells that the receipt of payment has been sent to the user's email id it is represented as *ReceiptSent*. To understand what happened in an interaction, we use the **uri** and **response_data** fields in the data. **uri** is the API endpoint from which the chatbot got the information to present the user. It consists of six different types :

- *api/chatbot/getMainOptions*
- *api/chatbot/getOptions*
- *api/chatbot/authCustomer*
- *api/chatbot/verifyCustomer*
- *api/chatbot/resendOtp*
- *api/chatbot/multiAction*

The **response_data** field consists of JSON data on information related to the interaction. The response data consists of some very useful fields, main amongst them being *messages* and *session_id*. *session_id* will be used further down the processing steps, but *messages* will be the key to understand what the interaction actually constituted. Except for *api/chatbot/multiAction*, all other **uri** and **response_data** have very few combination of *messages* which can be labelled by hand. For *multiAction*, there are many different possible values which the corresponding **response_data** can acquire. For this, initially extracting the text in the *messages* field of **response_data** which denotes what the chatbot output to the user, and used text clustering on it was done. But later, replacing all common tokens like phone numbers, email id, etc. and finding unique **messages**, which gave around 85 different types of *messages* was done. For these, a single short

string label was assigned which will be used for further analysis.

2.3 Parsing Engagements

Engagements are string of interactions which are part of the same session of back and forth between the chatbot and user. Converting the above interactions to engagement is necessary since we cannot decipher what the user was trying to achieve with the chatbot only by looking at single points of interactions, we need to look into groups of interactions to fully understand the outcome of the 'conversation' between the user and the chatbot. For example, if we just see an interaction where the user asks for bill details, we think that the user successfully acquired their bill details, but when we see the whole engagement, we see that after the user had asked for their bill details, the chatbot has thrown an error, which means that the outcome of the 'conversation' was unsuccessful. Thus engagements are a necessary preprocessing step for analysing the data

One issue with the raw data is that there is no direct data field which points to which interaction belongs to which engagement. Thus we need to look into the existing fields for a 'key' which points to which interaction belongs to which engagement. This 'key' is present in the form of *session_id*, which is a unique ID assigned to each interaction which identifies to which browsing session the interaction belonged to, which in turn means that it also points to which unique engagement the interaction belongs to. One thing to note is that *session_id* is not present in the same place for every interaction. For some interactions, it is present in the **response_data**, which was mentioned already in the previous section, but for some interactions it was present in the **headers** field, which contains other network metadata related to the interaction. Thus, we need to extract the *session_id* for each interaction based on where it is present. This is easy to do since both the fields are in JSON format which means the data will always be present in a fixed position which can be accessed by converting the JSON into a Python dictionary. Once we are able to get the *session_id*, now we need to know the order of the interaction in the engagement. For this we use the **time** field which contains the time at which the interaction is sent to the server. Since the engagement has to be linear in time, we use the *time* field to sort the interactions after grouping by *session_id*. This gives us the engagement in its complete form, which can be used for further analysis.

2.4 Parsing Multi-Engagements

Parsing engagements might look enough to analyse the data as a whole, but we need to consider something more regarding the engagements. Sometimes a user might log out of the session and then start another new session just after that since they needed a different type of issue to be handled by the chatbot, or they could also create a new session since they were not satisfied with the result of their engagement with the chatbot. These separate engagements can be combined into a new engagement called a **multi-engagement**, which consists of joining these unique engagements by same user separated only by a short amount of time. Multi-engagements are very useful to analyse as compared to simple single engagements, since they convey to us what the users true intention was when interacting with the chatbot which cannot be seen when these engagements are taken separately and analysed. Another use case, which will be seen later, is that this extends the number of **objectives** the user encountered with the chatbot, which improves our final analysis of the data. So the next step of data preprocessing is to combine engagements into multi-engagements and use these as the new engagements for the next steps.

Similar to engagement, a key is required to group the closely occurring engagements by same users into a multi-engagement. Here the key is two-fold, one is the user and another is the closeness of the engagements. For the first key, the situation is similar to *session_id*, albeit with another layer of difficulty. Unlike *session_id*, *account_no* is always present in **response_data**, but it is not present in all the interactions like *session_id*. This is because the *account_no* is only registered when the user logs in using their mobile. Thus, after an engagement is created, we need to scan throughout the engagement for *account_no* and then we propagate it all over the other interactions of the engagements. The next step involves finding closely occurring engagements. For this, we first group the engagements via *account_no* and find the difference in time between consecutive engagements (sorted via time). We then create a cumulative sum function which adds 1 when the time difference is greater than 5 minutes. This makes sure that engagements with time difference of less than 5 will be assigned the same value which can be used as a key for grouping. Thus, we use *account_no* as primary key and the cumulative sum as secondary key and grouping by the above keys gives us the multi-engagements.

One thing to note is that there are engagements which do not contain any *account_no*, since in those engagements the users did not login their account. The engagements which do not have any *account_no* are removed since they do not contain any useful information as the engagement usually stops at the login step or much before that. Thus for further analysis these types of engagements are removed and only multi-engagements which have *account_no* are used.

2.5 Parsing Objectives

Objectives represents the overall end result or the underlying of a continuous set of interactions. These objectives point to the reason behind what the user was trying to gain from their 'conversation' with the chatbot and we can apply many interesting statistical analysis on them to understand the behaviour of the users of the chatbot. An example of an objective could be solving an internet issue, where the user logs into the chatbot and tells the chatbot that they are facing an internet issue, which leads to that chatbot providing them with different troubleshooting steps and based on the user response, we can conclude that either the issue has been solved or the issue has not been solved and a ticket needs to be raised. This can be represented in form of a string of interactions as

$$\textit{ConfirmOpenURL} \rightarrow \textit{SelectIssue} \rightarrow \textit{TroubleshootingSteps} \rightarrow \textit{CheckIssueResolved} \rightarrow \textit{ProvideComments} \rightarrow \textit{TicketRaised}$$

We can label this as an objective, but there might be some more important interactions which can be added to this. One issue with finding objectives from the preprocessed data is that there is no concrete method for obtaining them like in the previous cases. We have to understand from the data what set of interactions could possibly form an objective. There can be various methods for doing this, but the method chosen in this case was to mine the most commonly occurring string of interactions from the data and use them as representations of objectives with minimal tweaking. Interactions which do not occur in the above common interactions are considered to be objectives onto themselves. There are many sequence mining algorithms present for this task, **PrefixSpan** was chosen since it is a very fast sequence mining algorithm and also has an easy implementation in Python.

2.5.1 PrefixSpan

PrefixSpan (**Prefix**-projected **S**equential **patt**er**n** mining) is a sequential pattern mining algorithm described in Pei *et al.* (2001), Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. It is a pattern-growth-based approach which works by recursively building prefix-based projections greater than the minimum support. A small example of the algorithm is as follows :

ID	Sequences
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

Table 2.1: An example list of sequences

Consider the set of sequences given in Table 2.1. We need to mine for subsequences with minimum support of 2 from the above list of sequences. First we find the support for each of the one length prefixes in the above sequences which is given as :

- <a> = 4
- <b> = 4
- <c> = 4
- <d> : 3
- <e> = 3
- <f> = 3
- <g> = 1

Since <g> has support less than 2, we ignore all occurrences of <g>. The prefix-based projection after ignoring all occurrences of <g> is given in Figure 2.1

<a>	<b>	<c>	<d>	<e>	<f>
<(abc)(ac)d(cf)>	<(_c)(ac)d(cf)>	<(ac)d(cf)>	<(cf)>	<(_f)(ab)(df)cb>	<(ab)(df)cb>
<(_d)c(bc)(ae)>	<(_c)(ae)>	<(bc)(ae)>	<c(bc)(ae)>	<(af)cbc>	<cbc>
<(_b)(df)cb>	<(df)cb>	<b>	<(_f)cb>		
<(_f)cbc>	<c>	<bc>			

Figure 2.1: Prefix projections for whole dataset

Now we recursively apply the algorithm by assuming each of the above prefix projection as a new sequence set. For the given example, let us continue with <d> prefix. The support for all one length subsequences is given as follows :

- <a> = 1
- <b> = 2
- <c> = 3
- <e> = 1
- <f> = 1
- <_f> = 1

We see that only <b> and <c> have a support greater than or equal to 2, thus we ignore all the other subsequences for the prefix projection which is given in Figure 2.2

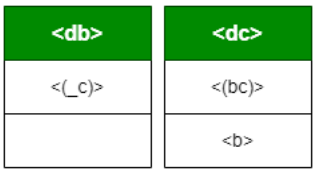

Figure 2.2: Prefix projections from projection of <d>

The above two projections shows us that we see that we cannot continue with <db>, since it has only one element which is less than the minimum support of 2. But <dc> can be extended to <dcb> which has a support of 2. Thus, from the recursive application on the projections of <d>, we get the mined subsequences as <d>, <db>, <dc> and <dcb>. We can mine more subsequences by recursively applying the PrefixSpan algorithm for the other projections too.

2.5.2 PrefixSpan for Objectives

Using PrefixSpan for finding Objectives is done via the following method

- First, we consider all the engagements (not multi-engagements, since it will be easier for processing) as sequence of interactions
- We apply PrefixSpan on the above sequences and mine important subsequences which contain interactions inter-related to each other
- Using the mined subsequences and some heuristics, we define some objective sequences which are large sequences, and any sequence is a subsequence of an objective sequence will be assigned the objective represented by the supersequence.

Using the above mined sequences, around 17 objectives have been defined. The list of

objectives is given below.

- Internet Issue
- Internet Issue Not Solved
- Internet Issue Solved
- Login Process
- Router Config
- Errors
- Request Already Raised
- Bill Details
- Payment Details
- Checking Details
- Data Package Details
- Receipt Details
- Reconnect Internet
- Open Service Requests
- Sent Credentials
- Ads Feedback
- No Open Complaints

2.6 Data Preprocessing Pipeline

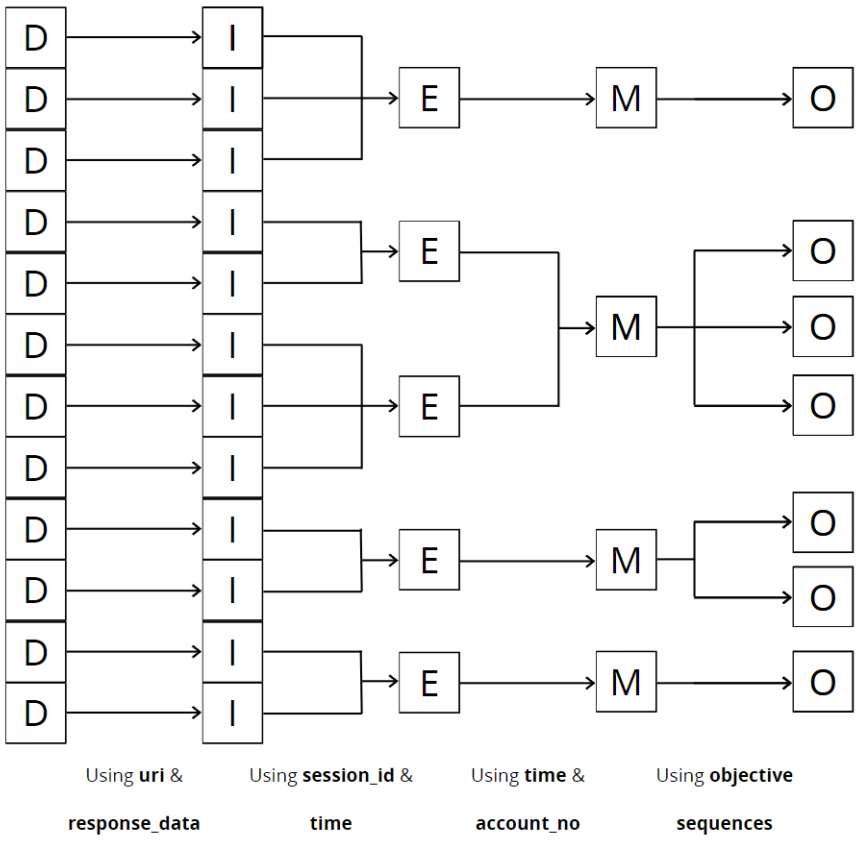

Figure 2.3: Data Pipeline for Preprocessing. D → Raw Data, I → Interaction, E → Engagement, M → Multi-Engagement, O → Objective

The data pipeline is built as follows :

- First, we read the **response_data** and **uri** to convert the raw data into a short string representation of the interaction.
- Next for each interaction, we get the *session_id* from either **response_data** or **headers** and sorting by **time**, we group the interactions into engagement.
- Next for each engagement we get the *account_no* and assign labels to each engagement by using a custom cumulative sum function which denotes which multi-engagement it belongs to. Using the former as primary key and latter as secondary key we group the engagements into multi-engagement.
- Next for each multi-engagement, we split the engagements by a particular interaction as separator, and from the resulting list of interaction sequences, we compare them to the objective sequence and assign the objective whose sequence the given interaction sequence is a subset of. This gives us sequences of objectives for each multi-engagement.

The data has been now preprocessed into a sequential format which can be used for further analysis.

CHAPTER 3

DATA ANALYSIS

From the resulting objectives, we look at the distribution of different objectives which is given in Table 3.1.

Objective	% Of Total Objectives
Internet Issue	67.8
Login Process	8.0
Router Config	5.5
Errors	5.2
Request Already Raised	4.8
Bill Details	3.1
Checking Details	2.2
Data Package Details	1.3
Receipt Details	0.7
Reconnect Internet	0.5
Open Service Requests	0.3
Sent Credentials	0.2
Ads Feedback	0.2
No Open Complaints	0.1
Payment Details	0.1

Table 3.1: Percentage Count of Different Objectives

From the data, we can see that *Internet Issue* constitutes around 70% of the interactions overall making it a huge majority compared to other objectives. Focusing on internet issues specifically can yield useful information regarding the user behaviour based on whether the issue have been solved by their engagement with the chatbot or not, thus the Internet Issue objective has been broken into three categories :

- *Internet Issue Solved*, which consists of Internet Issues which have been solved from the session with the chatbot
- *Internet Issue Not Solved*, which consists of Internet Issues which have not been solved by the session with the chatbot and a ticket had to be raised
- *Internet Issue*, which constitutes of Internet Issues whose result is unknown since the engagement ended with neither a ticket being raised or a interaction which told that the issue had been solved.

After splitting the *Internet Issue* objective into three parts, the percentage counts are given in Figure 3.1 :

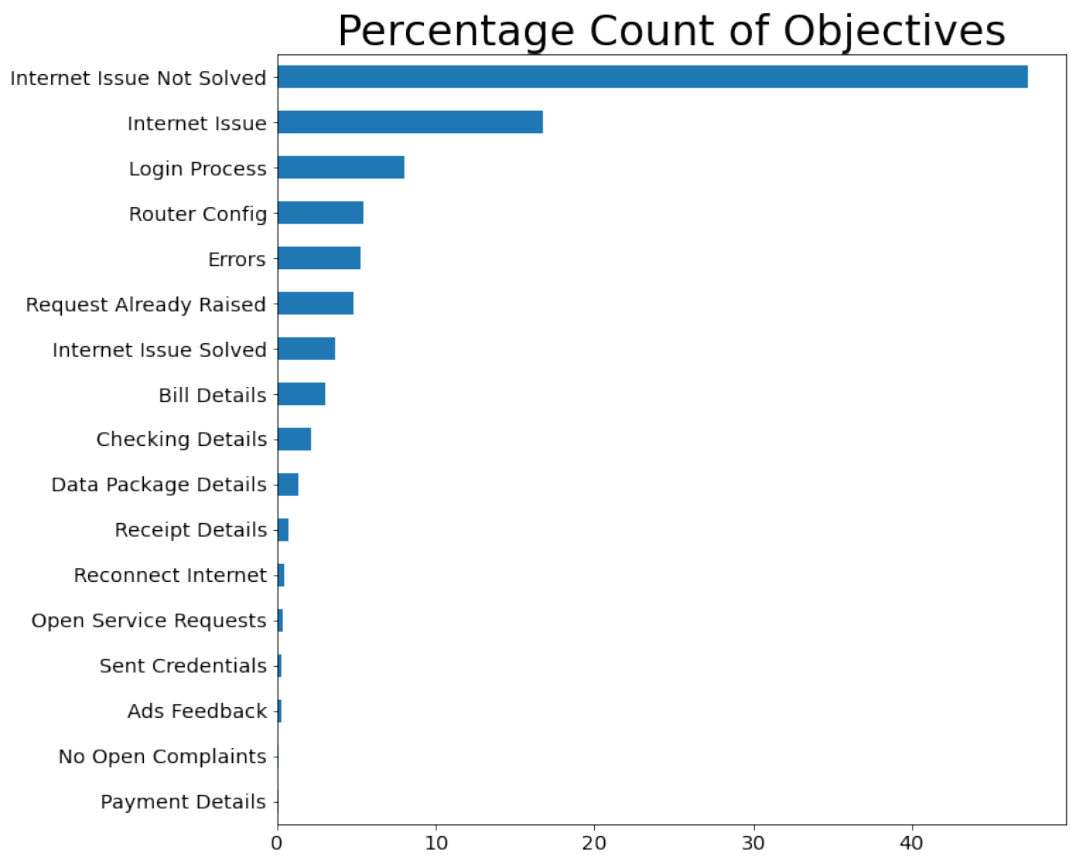

Figure 3.1: Percentage Count of Different Objectives

From the remaining objectives, the most relevant ones are :

- *Request Already Raised*, where an user is told their issue has a ticket already raised and is being looked on.
- *Router Config*, which is similar to Internet Issue, but specific to router issues like changing password and also has router specific troubleshooting
- *Bill Details*, where user requests for their internet bill to be displayed
- *Receipt Details*, where user requests for the receipts of their payments to be displayed

Other than the above objectives, all other objectives are either one-off (the engagements which contain them mostly do not contain any other objective) or statistically insignificant to consider.

3.1 Analysing Objectives as a Sequence

Due to the way objectives are created, they form a sequence for many engagements. This means that these objectives can be analysed as sequences themselves. This analysis can give us very strong insights on how users go from one type of objective to another based on their requirement. The most useful insight in this case would be to find which objective succeeds a given objective. This linear analysis can give us an idea of what type of behaviour the users exhibit when they interact with the chatbot. For this analysis, we model the objectives being generated from a **Hidden Markov Model (HMM)**, specifically a Multinomial HMM.

3.1.1 Hidden Markov Model

HMM is a generative probabilistic model, in which a sequence of observable variables $\mathbf{X}$ is generated by a sequence of internal hidden states $\mathbf{Z}$. The hidden states are not observed directly. The transitions between hidden states are assumed to have the form of a (first-order) Markov chain. They can be specified by the start probability vector π and a transition probability matrix A. The emission probability of an observable can be any distribution with parameters θ conditioned on the current hidden state. The HMM is completely determined by π, A and θ.

One of the problems involving modelling HMM sequences is that given a observed set of sequences, we need to find the above three given parameters to model the sequence as HMM. This can be solved via **Expectation-Maximization (EM)** algorithm. This is a iterative algorithm which alternates between the **Expectation(E)** step, which involves finding the expectation of the log-likelihood, which is calculated using the current set of parameters and the **Maximization(M)** step, which involves finding the parameters which maximizes the previously calculated expectation of the log-likelihood. This iteration is continued until the parameters settle on some stable values. The advantage of using EM algorithm is that it estimates both the Markov parameters (π and A) and also the latent emission parameters (θ) simultaneously.

3.1.2 Modelling Objectives as HMM

For our use case, we only require the transition matrix, since we only require to know which objectives succeed which ones. We fix the emission probabilities as each state only giving rise to one objective, that is

$$\theta_{S_i}(j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

where S_i denotes the state which only gives rise to objective i.

On modelling the list of objectives sequences by above method, we acquire a transition probability matrix which tells us which objective is most probable given the current objective. From the transition matrix, we look at the important objectives which we had pinpointed above and try to find which type of interactions lead to those and which type of interaction lead from those. Another reason why we ignore other objectives are that we cannot rely on the transition matrix generated for them as the amount of data we had for those objectives is very less, thus the model could have got stuck at a local minima for them. The most probable transitions along with their probabilities for important objectives are given below :

- Internet Issue $\rightarrow$
  - Internet Issue - 0.4
  - Internet Issue Not Solved - 0.5
- Internet Issue Not Solved $\rightarrow$
  - Request Already Raised - 0.9
- Internet Issue Solved $\rightarrow$
  - Internet Issue Not Solved - 0.58
  - Internet Issue - 0.28
- Request Already Raised $\rightarrow$
  - Request Already Raised - 0.8
- Router Config $\rightarrow$
  - Router Config - 0.5
  - Request Already Raised - 0.28

- Bill Details →
  - Error - 0.93
- Receipt Details →
  - Bill Details - 0.34
  - Receipt Details - 0.42

3.1.3 Recommendations

Based on the above transitions and some heuristics the following conclusions can be drawn :

- Internet Issue and Internet Issue Solved all lead to some form of Internet Issue again most of the time, which means that the users generally do not get their issue resolved even when they have confirmed that it is solved. This means that there is an issue in the flow where it is not confirmed from the user whether their issue has been resolved.
- Internet Issue Not Solved leads to Request Already Raised and Request Already Raised also leads to the same. This leads to the conclusion that although they have seen the message that a ticket had been raised regarding their issue, they do not seem to understand it and repeatedly mention that they have an internet issue.
- Router Config also has problems similar to Internet Issues, i.e. leading back to Router Issue and also to Request Already Raised mostly. This is expected since both are similar types of issues
- Trying to access Bill Details lead to Errors frequently, which does not happen often with other Objectives. Another interesting point to note is that many of these engagements where the user faces an error is usually followed by an engagement where the user accesses the Receipt Details.

Based on the above conclusions the following suggestions can be implemented in the chatbot :

- Before ending the chat session with the user who has complained about an internet issue, it should be confirmed with them whether their internet issue has been resolved and proceed based on the confirmation. As we see in the first observation, many users return back with an Internet Issue even after their previous engagement had been Internet Issue Solved, thus checking with the user before ending the chat session will be useful in reducing the number of times users interact with the chatbot.
- From the second observation, Internet Issue resolved consistently leads to users being told that their request is already being looked at, and even after being told that it is being looked at, they still register the Request Already Raised objective again. To prevent this, either of the following steps could be followed :

  - Display to the user the status of their ticket when they login to the chatbot initially if they have an existing ticket which is open when they open a chatbot session.
  - Point them to the Open Service Requests option if they have open tickets. This can be done by displaying it as the first option when user logs in to the chatbot or when they open the app, so that they can easily access to see the status of their ticket.
- We see that many of the users get confused on whether to access their bill or receipt details from the last observation. Although it is hard to pinpoint whether the reason behind it is due to lack of understanding the difference between both or lack of concentration while accessing the details, it would be helpful if the user is redirected towards their receipt details if they do not have any bills to check and vice versa.

3.2 Analysing User Responses

In Internet Issues Not Solved objective, there is an interaction called **ProvideComments** which asks for user to provide comments regarding their internet issue. Some examples of data in the interaction are :

- Internet is not working and high ping
- Lost internet connectivity
- Not working last 3 days
- Slow speed and sometimes getting disconnected

This interaction is helpful as it records the common reasons for users interacting with the chatbot and thus we can glean some useful information on where the chatbot is lacking which leads to the users to raise ticket for their issue.

3.2.1 Preprocessing Comments

The user comments is stored as a string which means we have to apply natural language processing techniques to analyze the data. After extracting all the comments which the users made in the ProvideComments section, we apply the following standard text preprocessing steps :

- **Case Normalization :** All the user text were made into lower case for uniformity sake. This ensures that there is no issues with words with same letters but different case are considered as different words.

- **Tokenization :** We split the text into words so that we can analyze the data based on the words contained in the text
- **Punctuation Removal :** We remove all kinds of punctuations from the text data since we only care about the words and not the sentences where punctuations are useful
- **Stopword Removal :** We remove stopwords since they occur in large numbers and provide very little information for analysis. However in our case, we do not remove the stopwords *not* and *no*, since there are many comments which contains phrases like *no internet* and *internet not working* which means that removing the above two stopwords will make the phrases lose their meaning
- **Lemmatization :** Lemmatization is the process of converting different inflected words into those words lemma, or dictionary form. This is useful in this case because words like 'connections' and 'routers' need to be converted to their singular form to better represent the terms for further analysis.

After applying the above basic preprocessing steps, we acquire the words which are part of the comments by the users as tokens. The first step would be to visualize the words in such a manner that we can pinpoint some important terms used by the users. For this, wordcloud representation of the terms is used. Wordcloud is a method of displaying words with their size given by the frequency with which they occur in the text. Figure 3.2 represents the wordcloud created from the user comments.

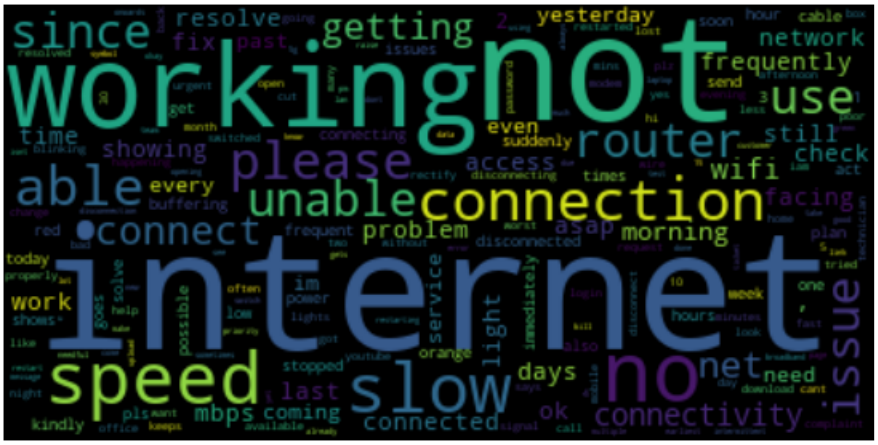

Figure 3.2: Wordcloud of user comments

We see that the most striking words from the wordcloud are 'internet', 'not', 'working', 'no', 'speed', connection', 'slow'. From this we can hypothesize that the common phrases which the users enter while they are facing internet issues are *internet not*

working, *no internet* and *slow speed* . To confirm this, we try to find the most occurring bigrams in the comments. The following Table 3.2 denotes the frequency of top ten bigrams in the user comments

Bigram	**Count**
not working	11057
internet not	8838
no internet	4153
not able	2195
use internet	1542
slow speed	1466
internet connection	1196
able use	1046
connect internet	805
working since	692

Table 3.2: Bigrams Count

For our working purposes, we can assume that *internet not working* means they have some kind of internet connection issue which causes no site to load, *no internet/no connection* means they are not getting the internet signal and *slow speed* means that their internet connection is slow. Although the second type may point to the first type for many cases in reality, for classification purposes they are considered to be different.

The next step would be to cluster these comments so that we can try to quantify how much of each type of entry is present in the data. For clustering the data, we will be using TF-IDF(Text Frequency - Inverse Document Frequency) representation of the user comments data and use K-Means to cluster the comments

3.2.2 TF-IDF

TF-IDF (Text Frequency - Inverse Document Frequency) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The main idea is that if a word or phrase appears frequently in an article and it is rarely found in other articles, it is considered that the word or phrase has a good class distinction capability and is suitable for classification. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. It is mainly composed of two parts, namely frequency of words and frequency of inverse texts.

Word frequency refers to the number of occurrences of a given word in the file. The inverse file frequency represents a measure of the general importance of a word. The frequency of the inverse of a word is divided by the total number of documents, which is divided by the number of documents containing the term, and then the resulting quotient logarithm. The formulas for word frequency (TF) and inverse text frequency (IDF) are as follows:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \qquad idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

- n_{ij} : Number of occurences of term t_i in file d_j
- $\sum_k n_{kj}$: Sum of occurences of all words in file d_j
- $|D|$: Total number of files in the corpus
- $\{j : t_i \in d_j\}$: Number of documents containing the word t_i

Thus, $tfidf_{ij} = tf_{ij} \cdot idf_j$ and is most of the times normalized by some function of tf and idf.

The reason why TF-IDF was chosen as the representation for clustering the comments are :

- It is easy to compute as there are many available libraries to compute it
- The order of the words are not that much necessary since our clustering mainly involves using some key words

With a proper min_{df} value, we can remove rarely occurring words, thus enabling us to only have the keywords which can be used for representing the data for clustering purpose.

3.2.3 Clustering Comments

We cluster the comments using the TF-IDF representation of the comments and using K-Means. The counts of different internet issues are plotted below in the pie chart 3.3:

Connection Issues involve comments which go along the lines of *internet not working*,

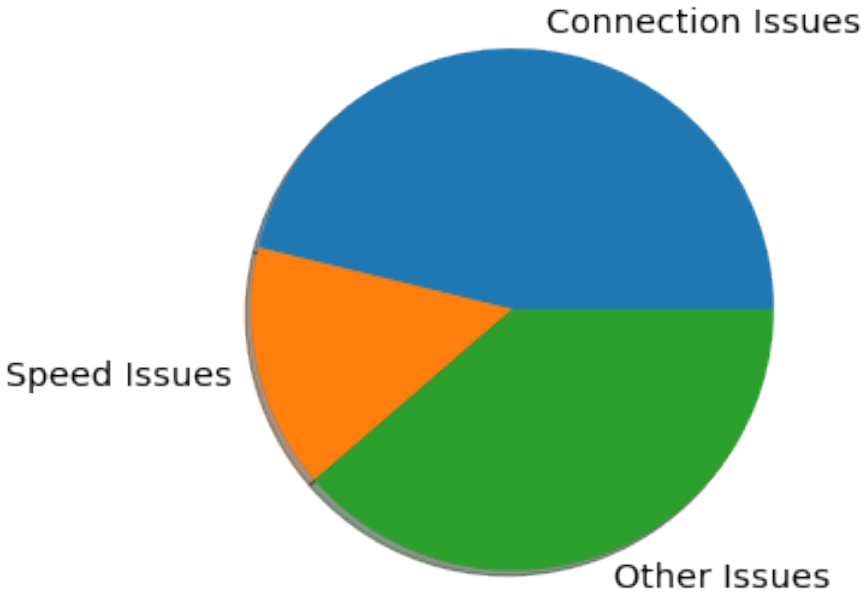

Figure 3.3: Count of Different Internet Issues

no internet, *no connection*, *unable to connect to internet* and so on. Speed Issues involve comments on the speed of the internet and Other Issues involve those which cannot be properly classified into the above two categories like intermittent issues, specific issues like not working in mornings, and so on.

Some observations which can be made regarding the above data :

- Most of the issues raised by the users via ticket are connection issues and comparatively less number of issues are speed based ones.
- Many of the comments in Other Issues are descriptive in nature about the particular issues faced like intermittent issues, internet not working at specific times of the day and so on, unlike Connection Issues where the comments are only about Internet not working.
- Although speed issue is common among the users, though begin lower in count compared to internet issue, speed issue as an option is not present in the list of internet issue provided in the chatbot.

From these observations, one can conclude that the variety of internet issues exists which are specific to the user and has to be looked into by technical people to solve the issue.

One suggestion which can help improve the user chatbot experience is by providing an option to separately raise a internet speed issue. There were no options in the chatbot which enables the user to specifically complain about speed issues, and having one with

appropriate troubleshooting steps could help reduce some of the users interaction time with the chatbot.

3.3 Extra Interactions

In some of the engagements, there are interactions which are repeated, i.e. the same interactions occur in the engagement one after another. An example is as follows :

Enter → NetIssue → CheckingConnectivity → ProvideComments → ProvideComments

We see that in the above engagement ProvideComments has been repeated twice in the end, which is used to get some information regarding what internet issue the user is facing currently.

The top two most repeated interactions in engagements are ConfirmOpenURL and ProvideComments. This is very interesting to note in the sense that those are both are usually one of the starting and ending interactions of Internet Issue group. The first argument which can be made is that this repetition is the highest simply because Internet Issues are statistically the highest occurring objectives out of all the objectives, thus these engagements which are part of the highest occurring objectives will be highest repeating too. But even if the above was the case, it is interesting to look at these interactions since they are the only interactions repeated in Internet Issues objectives and also they are the boundary interactions (occurs at the start and the end) of Internet Issues. There could be two reasons for being repeated :

- **Technical Error :** There could be some technical error in the implementation of these interactions that it sometimes cause the interactions to repeat if the users did something wrong. This has very low chance of being the case since all interactions seem to be coded very similar and these are the only two interactions which have this issue.
- **Human Hesitance :** The user could decide that they want to stop the engagement for some reason and try to start the chat again, but they inadvertently make some action which causes the chat to repeat the interaction with the user. This could be the more probable case for the chat repetition since the user can recognize that the engagement has not gone in the way they had liked at these interactions only.

The second case could be the most possible reason for the above repetitions, thus having a restart chat/stop chat option could be helpful in case people want to start over since

they feel that they have selected the wrong option to start with. This could help solve both the repetition of interactions and user trying to leave the chat in a proper manner.

CHAPTER 4

SUMMARY AND CONCLUSION

We needed to provide suggestions to improve the chatbot based on analysing the chat details between the user and chatbot. The data was provided in a raw JSON type format which had various fields. On close analysis, it was found that only a few of those fields contain data relevant for the analysis and thus these were retained and the rest were discarded. From looking at the data, we see that it is not in the format of chat sessions but only contained single interactions by the user. Also the chat were not in the format of natural language but based on options which was selected by the user. Thus, some preprocessing steps were required. These preprocessing steps generated different building blocks for structuring the data in an understandable way.

The first building block was interaction which consisted of one back and forth between the user and chatbot. This constitutes essentially a single datapoint in the raw data. To better represent the interaction, we used a short string representation which denotes what was the reason or the result of the single back and forth between the user and the chatbot. To find this short string representation, the fields uri and response_data were used. These two fields can be used to parse the short-string representation of the interaction.

The next building block which builds upon these interactions is engagement. Engagement denotes a string of interactions which are part of the same session of back and forth between the user and chatbot. This conversion from interactions to engagements is necessary since only then we will understand the result of what happened from the conversation between the user and chatbot. To group together the interactions into engagements, we need to find a key which can link these interactions. This key is session_id, which is present in either the response_data or headers. Using the session_id, we can group together the interactions into engagements.

Multi-engagements is the next building block, which which consists of unique engagements by same user separated only by a short amount of time. This is required to analyse more deeply how users interact with the chatbots and find links between each

interaction. To build multi-engagements from engagements, two levels of grouping was required, first from the user_id which had to be extracted and propagated throughout the engagements, and time, with a limit of 10 minutes between the engagements of same user for them to be considered as a part of a multi-engagement.

The final building block is objectives, which are a set of interactions which together convey either the intention of the user or result of the set of interactions. These objectives are not concrete and needed to be determined based on commonly occurring subsequence of interactions in engagements. Thus, common subsequences of interactions in engagements were mined using Prefix Span and based on analysing them, we were able to create objective sequences which consisted of sequences which point to an objective and any engagement which has any subsequence of this sequence will be assigned the objectives based on the order in which it appeared.

Now that we have the sequence of objectives for all the multi-engagements, we treat these sequences to be an Hidden Markov Model, and find the transition probabilities for the objectives, which gives us which engagement succeeds which ones frequently, thus giving us valuable information on the user behaviour.

Next, the comments of users who faced internet issues were analysed using natural language processing techniques. Basic NLP techniques like case normalization, tokenization, stopword removal and lemmatization were applied and the text was converted into a TF-IDF representation. The text were clustered and based on the results some useful insights were obtained. Also, many interactions being repeated in the start and end of some engagements which also raised some highlights regarding user behaviour.

Based on the above analysis, the following suggestions are made which can be implemented to improve the user experience with chatbot

- Before ending the chat session with the user who has complained about an internet issue, it should be confirmed with them whether their internet issue has been resolved and proceed based on the confirmation.
- To mitigate users from repeatedly trying to file internet issues on issues with ticket already raised, either of the following steps can be followed :
  - Display to the user the status of their ticket when they login to the chatbot initially if they have open tickets.
  - Point them to the Open Service Requests option if they have open tickets.
- Redirect users toward their receipt details if they do not have any bills to check

and vice versa

- Provide an option for users to raise internet speed issues, and if possible provide troubleshooting steps
- Provide a restart chat option especially while raising internet issues

Thus, using the user interaction data provided by ACT, areas of improvement for the chatbot have been found and appropriate suggestions provided. The above given suggestions can help improve user experience by reducing the number of engagement the user needs to have with the chatbot and can also improve user satisfaction fairly well. Chatbots are notoriously difficult to build since they require careful tuning based on where and for what purpose they are deployed, and in many cases, users eventually need warm body interaction to get their problems solved. But as chatbots continue to evolve with advancements in neural networks and in natural language processing, we also need to generally analyse the overall interactions between the users and chatbot and try to find business-level ideas which can be implemented in chatbot as opposed to technical ones which can improve user interactions effectively. One such project is the current one and hopefully such projects are undertaken to improve general user experience.

APPENDIX A

LIST OF INTERACTIONS

Given below is the list of interactions extracted from the dataset with a short string representation of what happened in the interaction :

- Enter
- ServeOptions
- AuthFail
- TicketRaised
- SelectIssue
- ProvideComments
- Relogin
- AuthOTP
- VerifySuccess
- NoActivePlans
- DueDateUnavailable
- DataUsed
- DueAmount
- NetIssue
- CheckingConnectivity
- CurrentPackage
- VerifyIncorr
- DueDate
- RetryLater
- MoreDetails
- FurtherQueries
- ReOTPSuccess
- Done
- NoBillDetails
- OpenServiceRequests
- SentCredentials
- NoReceipts
- TicketReopened
- VerifyError
- RequestAlreadyRaised
- AccountDisconnected
- SelectReceipt
- ReceiptSent
- SelectBill
- UnableToProcess
- PaymentDetails
- Ads
- ProvideFeedback
- ConfirmOpenURL
- CheckRouterOn
- PortStatus
- TroubleshootingSteps
- IssueResolved
- ReconnectInternet
- RestartRouter
- ServiceRestored
- AccountAlreadyActive
- CheckIssueResolved

- AccountReconnected
- PasswordUpdated
- EnterWifiPass
- WifiDetailsChanged
- VerifyFail
- CodeError
- NoOpenComplaints
- ReOTPFail
- Error
- ReopenedRequest
- PayBill
- SentBill
- AutoLogIn
- IsIssueResolved
- ConnectLAN
- PowerOnRouter
- ConfirmInternetPower
- ShareIssue
- NoIssue
- ConfirmIssueSites
- SelectRouterIssue
- SelectRouterModel
- CheckingRouterSettings
- ConfigGuide

APPENDIX B

LIST OF OBJECTIVES

Given below are the list of objectives with their corresponding objective sequence. As mentioned, the one-off interactions are also mentioned as objectives onto themselves like No Open Complaints and other such interactions.

Login Process

AuthFail → AuthOTP → ReOTPFail → ReOTPSuccess → VerifyIncorr → VerifyFail → VerifySuccess → VerifyError

Internet Issue

ConfirmOpenURL → SelectIssue → ShareIssue → TroubleshootingSteps → PortStatus → RestartRouter → ConfirmInternetPower → ConfirmIssueSites → AutoLogIn → ConnectLAN → IssueResolved → ServiceRestored → NoIssue → CheckIssueResolved → ProvideComments → TicketRaised → TicketReopened → ReopenedRequest

Router Config

SelectRouterIssue → EnterWifiPass → CheckRouterOn → EnterWifiPass → RestartRouter → PowerOnRouter → WifiDetailsChanged → CheckingRouterSettings → ConfirmOpenURL → SelectRouterModel → ConfigGuide → ServiceRestored → PasswordUpdated → CheckIssueResolved → ProvideComments → TicketRaised

Current Package

CurrentPackage → DueAmount → Done

Ads Feedback

Ads → ProvideFeedback

Data Package Details

CurrentPackage → NoActivePlans → DueDate → DueDateUnavailable → DueAmount → DataUsed → Done

Checking Details

MoreDetails → FurtherQueries

Bill Details

NoBillDetails → SelectBill → SentBill

Receipt Details

NoReceipts → SelectReceipt → ReceiptSent

Reconnect Internet

ReconnectInternet → Ads → AccountAlreadyActive → AccountDisconnected → AccountReconnected → PayBill

Errors

RetryLater → UnableToProcess → Error → Relogin → CodeError

Sent Credentials

SentCredentials

Request Already Raised

RequestAlreadyRaised

Open Service Requests

OpenServiceRequests

Payment Details

PaymentDetails

No Open Complaints

NoOpenComplaints

REFERENCES

1. **Følstad, A.** and **C. Taylor** (2021). Investigating the user experience of customer service chatbot interaction: a framework for qualitative analysis of chatbot dialogues. *Quality and User Experience*, **6**, 1–17.

2. **Thorat, S. A.** and **V. Jadhav** (2020). A review on implementation issues of rule-based chatbot systems. *Social Science Research Network*.

3. **Efraim, O.**, **V. Maraev**, and **J. Rodrigues**, Boosting a rule-based chatbot using statistics and user satisfaction ratings. 2017.

4. **Peng, Z.** and **X. Ma** (2019). A survey on construction and enhancement methods in service chatbots design. *CCF Trans. Pervasive Comput. Interact.*, **1**, 204–223.

5. **Fournier Viger, P.**, **C.-W. Lin**, **U. Rage**, **Y. S. Koh**, and **R. Thomas** (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, **1**, 54–77.

6. **Pei, J.**, **J. Han**, **B. Mortazavi-Asl**, **H. Pinto**, **Q. Chen**, **U. Dayal**, and **M. Hsu**, Prefixspan: Mining sequential patterns by prefix-projected growth. *In Proceedings of the 17th International Conference on Data Engineering*. IEEE Computer Society, USA, 2001. ISBN 0769510019.

7. **Bilmes, J. A.** (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *CTIT technical reports series*.

8. **Rabiner, L.** (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–286.

9. **Liu, C.-z.**, **Y.-x. Sheng**, **Z.-q. Wei**, and **Y.-Q. Yang**, Research of text classification based on improved tf-idf algorithm. *In 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*. 2018.

10. **Gao, C.** (2018–). PrefixSpan, BIDE, and FEAT in Python 3. URL `https://pypi.org/project/prefixspan/`.

11. **hmmlearn developers** (2010–). Hmm tutorial and documentation. URL `https://hmmlearn.readthedocs.io/en/latest/tutorial.html`.

12. **Andreas Mueller** (2020–). Wordcloud for python. URL `https://amueller.github.io/word_cloud/`.

13. **scikit-learn developers** (2007–). Clustering text documents using k-means. URL `https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html`.