0  **PHASE - 1**

**Problem definition and design thinking:**

In this part we need to understand the problem statement and create a document on what we have understood and how we will proceed ahead with solving the problem.

**Problem Definition:**

⮚ The chatbot should solve the inconvenience experience among the users.

⮚ By providing the helpful information, answer frequently asked questions (FAQs), and offer a satisfactory experience.

**Objectives:**

⮚ We aim to achieve

o improving user engagement

o providing instant support

o automating tasks.

o Solving common queries

o Resolve user pain points

**1. Personal Design:**

A chatbot person is like the soul of our chatbot, a carefully crafted character that embodies the tone, voice, and personality of your [virtual assistant](https://botpenguin.com/chatbot-vs-virtual-assistant-know-your-basics/).

⮚ Name – Ibgram

⮚ Voice – Male/Female (john/medona)

⮚ Tone - Flexible

⮚ Communication style - Assertive

**2. User Scenarios:**

⮚ When the users are interacting with the chatbot.

⮚ The frequent questions asked by the users like forgot password.

⮚ Conversation help

⮚ Feedback of the chatbot experience

**Conversation Flow:**

By identifying the user query with the keyword recognition to provide appropriate and relevant suggested answers to the users with the help of IBM watson.

**Response Configuration:**

The response of the chatbot is under three stages of processing

o Intent detection – to identify the overall idea of the query.

o Entity detection – to identify the keyword of the query and fetch sub parts of the query.

o Irrelevant detection – to identify the query given by the user is query or not.

o Auto error correction – to auto correct the human errors. (forgot passwordèforgot password)

**Platform Integration:**

To integrate with the other popular platforms similar to us like

o Facebook messenger

o Slack

To improve the efficiency and performance of the chatbot and identify the user type.

**User Experience:**

To know the experience by taking the feedback survey after the end of the conversation.

5 star-excellent,4star-good,3-satisfactory,2-slightly poor,1-poor.

--------------------------------------------------------------------------------------

**PHASE - 2**

**Innovation:**

ChatGPT is an innovative language model powered by Artificial Intelligence that can be used for documentation purposes. It can generate content for a wide range of subjects and provide coherent and contextually relevant answers. With its deep learning techniques and extensive training on a diverse dataset from the Internet, ChatGPT can assist in creating informative and engaging documentation. Whether you need assistance with technical writing, user manuals, or any other form of documentation, ChatGPT can help provide valuable content to enhance your documentation efforts.

1.Natural Language Processing (NLP): You can use GPT-3 to improve NLP tasks, such as sentiment analysis, text summarization, and language translation. By integrating GPT-3 into your Python-based NLP application, you can enhance the quality of these tasks.

2. Chatbots and Virtual Assistants: Integrate GPT-3 into your chatbot or virtual assistant to provide more natural and context-aware responses. You can use the OpenAI GPT-3 API to generate conversational responses.

3.Content Generation: Use GPT-3 to automatically generate content for your website, blog, or social media. Python can be used to create a system that leverages GPT-3 to produce high-quality articles, product descriptions, or marketing copy.

4.Code Generation: You can utilize GPT-3 to help generate code snippets, document code, or assist with debugging. For example, you can create a Python script that takes a natural language code request and converts it into code.

5.Data Analysis and Reporting: Enhance data analysis and reporting tools by integrating GPT-3. It can help generate descriptive insights and explanations from data, making reports more understandable.

6.Customer Support and Helpdesks: Improve customer support by integrating GPT-3 into your support ticket system. GPT-3 can suggest solutions or answer common customer queries, reducing response time and improving efficiency.

7.Language Translation: You can build a language translation tool using GPT-3 for translating text or speech in real-time. This can be particularly useful for applications requiring multilingual support.

8.Content Personalization: Utilize GPT-3 to personalize content recommendations on your website or app.

9.Gaming and Interactive Storytelling: Create interactive and dynamic storytelling in games or applications by using GPT-3 to generate plotlines, dialogues, and character interactions.

10.Medical and Legal Assistants: Integrate GPT-3 into applications designed for the medical or legal fields to help generate reports, summaries, and recommendations based on patient or case information.

------------------------------------------------------------------------------------------------------

**PHASE - 3**

**Model Chabot using python:**

**Introduction:**

Creating a Chabot in Python involves several steps, and you can choose different approaches depending on your specific requirements.

We provide with a simple example using Python and the NLTK library for natural language processing.

This Chabot will respond to a few predefined questions.

For a more advanced Chabot, we would typically use machine learning techniques and more extensive datasets.

**Step1:** First, we will make include NLTK installed by using install it pip:

Command:

Pip install nltk

**Step2:** Now, let's create a basic Chabot:

import nltk

from tk.chat.util import Chat, reflections

# Define patterns and responses for the chatbot

r'hello|hi|hey', ['Hello!', 'Hi there!']),

(how are you', ['I am a chatbot, I do not have feelings, but thanks for asking!']),

(r'what is your name', ['I am a chatbot. You can call me ChatGPT.']),

(r'(.\*) age', ['I do not have an age.']),

(r'(.\*) (created|made)', ['I was created by OpenAI using GPT-3.5.']),

(r'(.\*)', ['I am not sure I understand.']),

]

# Define some reflection options to make the conversation more natural

reflections = {

"am": "are",

"was": "were",

"I": "you",

"I’d": "you would",

( "I’ve": "you have",

"I’ll": "you will",

"my": "your",

"are": "am",

"you're": "I'm",

"you've": "I have",

"you'll": "I will",

"your": "my",

"yours": "mine",

"you": "me",

"me": "you",

}

# Create a chatbot using the patterns and reflections

chatbot = Chat(patterns, reflections)

# Start the conversation

print("Hello! I'm a simple chatbot. You can start a conversation with me. Type 'quit' to end the chat.")

while True:

user input = input("You: ")

if user\_input.lower() == 'quit':

print("Chatbot: Goodbye!")

break

response = chatbot.respond(user input)

print("Chatbot: " + response)

**Step3:**

In this example, we defined some patterns and responses for the Chabot.

You can customize these patterns and responses to fit your needs.

The Chabot uses reflection to make the conversation more natural by transforming pronouns.

**Step4:**

To end the conversation, type "quit" as the user input.

**Model Chabot using python:**

**Introduction:**

Creating a Chabot in Python involves several steps, and you can choose different approaches depending on your specific requirements.

We provide with a simple example using Python and the NLTK library for natural language processing.

This Chabot will respond to a few predefined questions.

For a more advanced Chabot, we would typically use machine learning techniques and more extensive datasets.

**Step1:** First, we will make include NLTK installed by using install it pip:

Command:

Pip install nltk

**Step2:** Now, let's create a basic Chabot:

import nltk

from nltk.chat.util import Chat, reflections

# Define patterns and responses for the chatbot

patterns = [

(r'hello|hi|hey', ['Hello!', 'Hi there!']),

(how are you', ['I am a chatbot, I do not have feelings, but thanks for asking!']),

(r'what is your name', ['I am a chatbot. You can call me ChatGPT.']),

(r'(.\*) age', ['I do not have an age.']),

(r'(.\*) (created|made)', ['I was created by OpenAI using GPT-3.5.']),

(r'(.\*)', ['I am not sure I understand.']),

]

# Define some reflection options to make the conversation more natural

reflections = {

"am": "are",

"was": "were",

"I": "you",

"I’d": "you would",

"I’ve": "you have",

"I’ll": "you will",

"my": "your",

"are": "am",

"you're": "I'm",

"you've": "I have",

"you'll": "I will",

"your": "my",

"yours": "mine",

"you": "me",

"me": "you",

}

# Create a chatbot using the patterns and reflections

chatbot = Chat(patterns, reflections)

# Start the conversation

print("Hello! I'm a simple chatbot. You can start a conversation with me. Type 'quit' to end the chat.")

while True:

user input = input("You: ")

if user\_input.lower() == 'quit':

print("Chatbot: Goodbye!")

break

response = chatbot.respond(user input)

print("Chatbot: " + response)

**Step3:**

In this example, we defined some patterns and responses for the Chabot.

You can customize these patterns and responses to fit your needs.

The Chabot uses reflection to make the conversation more natural by transforming pronouns.

**Step4:**

To end the conversation, type "quit" as the user input.This is basic Chabot. For more advanced Chabot’s with natural language understanding, we will typically use machine learning techniques like neural networks, and it might consider frameworks like Rasa or the OpenAI GPT-3 API for more sophisticated interactions.

**DEMO IMAGE:**

-----------------------------------------------------------------------------------------------------------------------------------------------

**PHASE - 4**

**CREATE A CHATBOT**

**DEFINITION:**

Rule based chat bot follow a set of predefined rules to determine the correct response to user’s input. Self learning chat bot uses machine learning algorithms to understand and respond to user input. Self learning chat bot can be further divided into Retrieval based and Generative based chat bot.

In this sense ,it is presented an inceptive theoretical framework through a formal way for chat bot that can be used as a reference chat bot that can be used as to reference explore, compose, build and discuss with chat bot.

**DatasetLink: https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot**

**CODE:**

import pandas as pd

import numpy as np

import string

from string import digits

import matplotlib.pyplot as plt

import seaborn as sns

import re

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from keras.layers import Input, LSTM, Embedding, Dense, Bidirectional, Concatenate, Dot, Activation, TimeDistributed

from keras.models import Model

from keras.utils import plot\_model

def data\_stats(lines, input\_tok\_split\_fn, target\_tok\_split\_fn):

input\_tokens=set()

for line **in** lines.input:

for tok **in** input\_tok\_split\_fn(line):

if tok **not** **in** input\_tokens:

input\_tokens.add(tok)

target\_tokens=set()

for line **in** lines.target:

for tok **in** target\_tok\_split\_fn(line):

if tok **not** **in** target\_tokens:

target\_tokens.add(tok)

input\_tokens = sorted(list(input\_tokens))

num\_encoder\_tokens = len(input\_tokens)

num\_decoder\_tokens = len(target\_tokens)

max\_encoder\_seq\_length = np.max([len(input\_tok\_split\_fn(l))

for l **in** lines.input])

max\_decoder\_seq\_length=np.max([len(target\_tok\_split\_fn(l))

for l **in** lines.target])

return input\_tokens, target\_tokens, num\_encoder\_tokens, num\_decoder\_tokens,max\_encoder\_seq\_length,max\_decoder\_seq\_length

Linkcode

print('Number of samples:', len(lines))

print('Number of unique input tokens:' , num \_ encoder\_tokens)

print('Numberofuniqueoutputtokens:',num\_decoder\_num\_decoder

\_tokens)

print('Max sequence length for inputs:', max\_encoder\_seq\_length)

print('Maxsequencelengthfor outputs:', max\_decoder\_seq\_length)

**SAMPLE INPUT**:

⮚ hi, how are you doing?

⮚ i'm fine. how about yourself?

⮚ i'm fine. how about yourself?

⮚ i'm pretty good. thanks for asking.

⮚ where are you going to school? i'm going to pcc.

⮚ how do you like it so far?

⮚ i like it so far. my classes are pretty good right now.

⮚ it doesn't look very nice outside today.

⮚ you're right. i think it's going to rain later.

**BUILDING THE CHAT BOT BY INTEGRETING IT INTO A WEB APP USING FLASH:**

Step1: Set up your Open AI account. ...

Step 2: Set Up Your Environment. ...

Step 3: Install Required Libraries. ...

Step 4: Create Flask App Structure. ...

Step 5: Create HTML Template: ...

Step 6: Implement Backend Logic: ...

Step 7: Run the Flask App:

**SAMPLE OUTPUT:**

**CONCLUSION**:

From my perspective, chat bot or smart assistance with artificial intelligence are dramatically changing businesses. There is a wide range of chat bot building platform that are available for various enter prices.

Chat bot can reach out to a Large audience on messaging apps and be more effective than humans.

-------------------------------------------------------------------------

**PHASE - 5**

**Dataset**:

Another great way to collect data for your chatbot development is through mining words and utterances from your existing human-to-human chat logs. You can search for the relevant representative utterances to provide quick responses to the queries.

**DatasetLink: https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot**

**Code**:

Filter changed files

21 changes: 19 additions & 2 deletions21

chatterbot/comparisons.py

@@ -2,6 +2,7 @@

This module contains various text-comparison algorithms This module contains various text-comparison algorithms

designed to compare one statement to another. designed to compare one statement to another.

""" """

from chatterbot.exceptions import OptionalDependencyImportError

from difflib import SequenceMatcher from difflib import SequenceMatcher

@@ -63,7 +64,15 @@ class SpacySimilarity(Comparator):

def \_\_init\_\_(self, language): def \_\_init\_\_(self, language):

super().\_\_init\_\_(language) super().\_\_init\_\_(language)

import spacy try:

import spacy

except ImportError:

message = (

'Unable to import "spacy".\n'

'Please install "spacy" before using the SpacySimilarity comparator:\n'

'pip3 install "spacy>=2.1,<2.2"'

)

raise OptionalDependencyImportError(message)

self.nlp = spacy.load(self.language.ISO\_639\_1) self.nlp = spacy.load(self.language.ISO\_639\_1)

@@ -108,7 +117,15 @@ class JaccardSimilarity(Comparator):

def \_\_init\_\_(self, language): def \_\_init\_\_(self, language):

super().\_\_init\_\_(language) super().\_\_init\_\_(language)

import spacy try:

import spacy

except ImportError:

message = (

'Unable to import "spacy".\n'

'Please install "spacy" before using the JaccardSimilarity comparator:\n'

'pip3 install "spacy>=2.1,<2.2"'

)

raise OptionalDependencyImportError(message)

self.nlp = spacy.load(self.language.ISO\_639\_1) self.nlp = spacy.load(self.language.ISO\_639\_1)

21 changes: 19 additions & 2 deletion 21

chatterbot/corpus.py

@@ -1,11 +1,18 @@

import os import os

import io import io

import glob import glob

import yaml from pathlib import Path

from chatterbot.exceptions import OptionalDependencyImportError

try: try:

from chatterbot\_corpus.corpus import DATA\_DIRECTORY from chatterbot\_corpus.corpus import DATA\_DIRECTORY

except (ImportError, ModuleNotFoundError): except (ImportError, ModuleNotFoundError):

DATA\_DIRECTORY = os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)), 'data') # Default to the home directory of the current user

DATA\_DIRECTORY = os.path.join(

Path.home(),

'chatterbot\_corpus',

'data'

)

CORPUS\_EXTENSION = 'yml' CORPUS\_EXTENSION = 'yml'

@@ -38,6 +45,16 @@ def read\_corpus(file\_name):

""" """

Read and return the data from a corpus json file. Read and return the data from a corpus json file.

""" """

try:

import yaml

except ImportError:

message = (

'Unable to import "yaml".\n'

'Please install "pyyaml" to enable chatterbot corpus functionality:\n'

'pip3 install pyyaml'

)

raise OptionalDependencyImportError(message)

with io.open(file\_name, encoding='utf-8') as data\_file: with io.open(file\_name, encoding='utf-8') as data\_file:

return yaml.safe\_load(data\_file) return yaml.safe\_load(data\_file)5 changes: 5 additions & 0 deletions5

chatterbot/exceptions.py

@@ -0,0 +1,5 @@

class OptionalDependencyImportError(ImportError):

"""

An exception raised when a feature requires an optional dependency to be installed.

"""

pass

11 changes: 10 additions & 1 deletion11

chatterbot/logic/time\_adapter.py

@@ -1,6 +1,7 @@

from datetime import datetime from datetime import datetime

from chatterbot.logic import LogicAdapter from chatterbot.logic import LogicAdapter

from chatterbot.conversation import Statement from chatterbot.conversation import Statement

from chatterbot.exceptions import OptionalDependencyImportError

class TimeLogicAdapter(LogicAdapter): class TimeLogicAdapter(LogicAdapter):

@@ -18,7 +19,15 @@ class TimeLogicAdapter(LogicAdapter):

def \_\_init\_\_(self, chatbot, \*\*kwargs): def \_\_init\_\_(self, chatbot, \*\*kwargs):

super().\_\_init\_\_(chatbot, \*\*kwargs) super().\_\_init\_\_(chatbot, \*\*kwargs)

from nltk import NaiveBayesClassifier try:

from nltk import NaiveBayesClassifier

except ImportError:

message = (

'Unable to import "nltk".\n'

'Please install "nltk" before using the TimeLogicAdapter:\n'

'pip3 install nltk'

)

raise OptionalDependencyImportError(message)

self.positive = kwargs.get('positive', [ self.positive = kwargs.get('positive', [

'what time is it', 'what time is it',

13 changes: 11 additions & 2 deletion 13

chatterbot/logic/unit\_conversion.py

@@ -1,5 +1,6 @@

from chatterbot.logic import LogicAdapter from chatterbot.logic import LogicAdapter

from chatterbot.conversation import Statement from chatterbot.conversation import Statement

from chatterbot.exceptions import OptionalDependencyImportError

from chatterbot import languages from chatterbot import languages

from chatterbot import parsing from chatterbot import parsing

from mathparse import mathparse from mathparse import mathparse

@@ -22,7 +23,15 @@ class UnitConversion(LogicAdapter):

def \_\_init\_\_(self, chatbot, \*\*kwargs): def \_\_init\_\_(self, chatbot, \*\*kwargs):

super().\_\_init\_\_(chatbot, \*\*kwargs) super().\_\_init\_\_(chatbot, \*\*kwargs)

from pint import UnitRegistry try:

from pint import UnitRegistry

except ImportError:

message = (

'Unable to import "pint".\n'

'Please install "pint" before using the UnitConversion logic adapter:\n'

'pip3 install pint'

)

raise OptionalDependencyImportError(message)

self.language = kwargs.get('language', languages.ENG) self.language = kwargs.get('language', languages.ENG)

self.cache = {} self.cache = {}

@@ -82,7 +91,7 @@ def get\_unit(self, unit\_variations):

def get\_valid\_units(self, from\_unit, target\_unit): def get\_valid\_units(self, from\_unit, target\_unit):

""" """

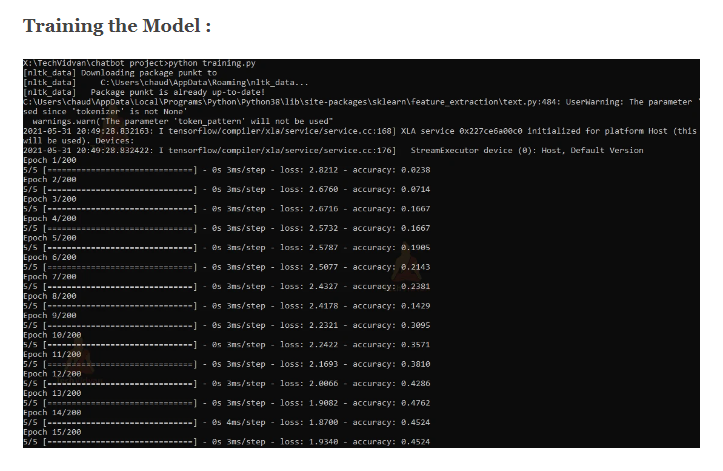
Returns the firt match `pint.unit.Unit` object for from\_unit and Returns the first match `pint.unit.Unit` object for from\_unit and

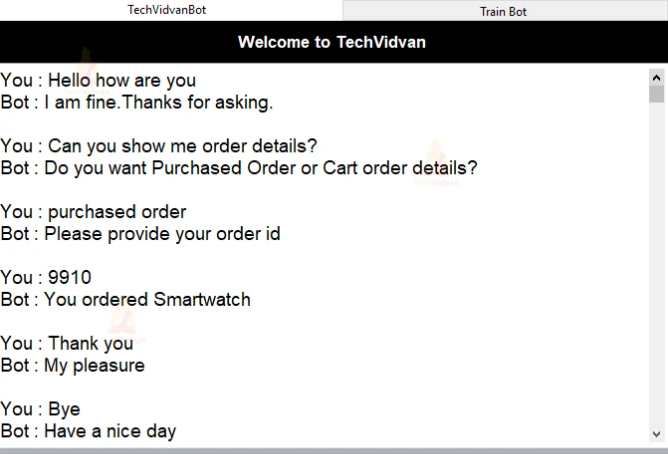
target\_unit strings from a possible variation of metric unit names target\_unit strings from a possible variation of metric unit names

**SAMPLE INPUT STATEMENT:**

hi, how are you doing? i'm fine. how about yourself? i'm fine. how about yourself? i'm pretty good. thanks for asking. i'm pretty good. thanks for asking. no problem. so how have you been? no problem. so how have you been? i've been great. what about you? i've been great. what about you? i've been good. i'm in school right now. i've been good. i'm in school right now. what school do you go to? what school do you go to? i go to pcc. i go to pcc. do you like it there? do you like it there? it's okay. it's a really big campus. it's okay. it's a really big campus. good luck with school. good luck with school. thank you very much. how's it going? i'm doing well. how about you? i'm doing well. how about you? never better, thanks. never better, thanks. so how have you been lately? so how have you been lately? i've actually been pretty good. you? i've actually been pretty good. you? i'm actually in school right now. i'm actually in school right now. which school do you attend? which school do you attend? i'm attending pcc right now. i'm attending pcc right now. are you enjoying it there? are you enjoying it there? it's not bad. there are a lot of people there. it's not bad. there are a lot of people there. good luck with that. good luck with that. thanks. how are you doing today? i'm doing great. what about you? i'm doing great. what about you? i'm absolutely lovely, thank you. i'm absolutely lovely, thank you. everything's been good with you? everything's been good with you? i haven't been better. how about yourself? i haven't been better. how about yourself? i started school recently. i started school recently. where are you going to school? where are you going to school? i'm going to pcc. i'm going to pcc. how do you like it so far? how do you like it so far? i like it so far. my classes are pretty good right now. i like it so far. my classes are pretty good right now. i wish you luck. it's an ugly day today. i know. i think it may rain. i know. i think it may rain. it's the middle of summer, it shouldn't rain today. it's the middle of summer, it shouldn't rain today. that would be weird. that would be weird. yeah, especially since it's ninety degrees outside. yeah, especially since it's ninety degrees outside. i know, it would be horrible if it rained and it was hot outside. i know, it would be horrible if it rained and it was hot outside. yes, it would be. yes, it would be. i really wish it wasn't so hot every day. i really wish it wasn't so hot every day. me too. i can't wait until winter. me too. i can't wait until winter. i like winter too, but sometimes it gets too cold. i like winter too, but sometimes it gets too cold. i'd rather be cold than hot. i'd rather be cold than hot. me too. it doesn't look very nice outside today. you're right. i think it's going to rain later. you're right. i think it's going to rain later. in the middle of the summer, it shouldn't be raining. in the middle of the summer, it shouldn't be raining. that wouldn't seem right. that wouldn't seem right. considering that it's over ninety degrees outside, that would be weird. considering that it's over ninety degrees outside, that would be weird. exactly, it wouldn't be nice if it started raining. it's too hot. exactly, it wouldn't be nice if it started raining. it's too hot. i know, you're absolutely right. i know, you're absolutely right. i wish it would cool off one day. i wish it would cool off one day. that's how i feel, i want winter to come soon. that's how i feel, i want winter to come soon. i enjoy the winter, but it gets really cold sometimes. i enjoy the winter, but it gets really cold sometimes. i know what you mean, but i'd rather be cold than hot. i know what you mean, but i'd rather be cold than hot. that's exactly how i feel. i wish it was a nicer day today. that is true. i hope it doesn't rain. that is true. i hope it doesn't rain. it wouldn't rain in the middle of the summer. it wouldn't rain in the middle of the summer. it wouldn't seem right if it started raining right now. it wouldn't seem right if it started raining right now. it would be weird if it started raining in ninety degree weather. it would be weird if it started raining in ninety degree weather. any rain right now would be pointless. any rain right now would be pointless. that's right, it really would be. that's right, it really would be. i want it to cool down some. i want it to cool down some. i know what you mean, i can't wait until it's winter. i know what you mean, i can't wait until it's winter. winter is great. i wish it didn't get so cold sometimes though. winter is great. i wish it didn't get so cold sometimes though. i would rather deal with the winter than the summer. it's such a nice day. yes, it is. yes, it is.

**SAMPLE OUTPUT:**



sdfwefgergrgger

et

**CONCLUSION:**

In this project, we have introduced a Chabot that is able to interact with users. This Chabot can answer queries in the textual user input. For this purpose, AIML with program-o has been used. The Chabot can answer only those questions which he has the answer in its AIML dataset. So, to increase the knowledge of the chatbot, we can add the APIs of Wikipedia, Weather Forecasting Department, Sports, News, Government and a lot more. In such cases, the user will be able to talk and interact with the chatbot in any kind of domain. Using APIs like Weather, Sports, News and Government Services, the chatbot will be able to answer the questions outside of its dataset and which are currently happening in the real world. The next step towards building chatbots involves helping people to facilitate their work and interact with computers using natural language or using their set of rules. Future Such chatbots, backed by machine-learning technology, will be able to remember past conversations and learn from them to answer new ones. The challenge would be conversing with the various multiple bot users and multiple users. As future work, we can make a chatbot that is based on AIML and LSA. This technology will enable a client to interact with a chatbot in a more natural fashion. We can enhance the discussion by including and changing patterns and templates for general client queries using AIML and the right response are given more often than LSA.