TASK:9

To Build an Intelligent **Chatbot system** with Python and Dialog-flow using Interactive Text Mining Framework for Exploration of Semantic Flows in Large Corpus of Text.

To Build an Intelligent Chatbot system with Python and Dialog-flow using Interactive Text Mining Framework for Exploration of Semantic Flows in Large Corpus of Text.
CO4   S3

- To integrate with Google Cloud Speech-to-Text and third-party services such as Google Assistant, Amazon Alexa, and Facebook Messenger.
- Configure Dialogflow to manage your data across GCP services and let you optionally integrate Google Assistant.

Tools- Python, Dialog-flow Framework

TO BUILD AN INTELLIGENT **CHATBOT SYSTEM** WITH PYTHON AND DIALOG-FLOW USING INTERACTIVE TEXT MINING FRAMEWORK FOR EXPLORATION OF SEMANTIC FLOWS IN LARGE CORPUS OF TEXT

AIM:

To build an intelligent chatbox system with Python and dialog-flow using interactive text mining framework for exploration of semantic flow in large corpus of Text

ALGORITHM:

Steps to create an intelligent chatbot using OpenAI APIs:

1. Sign up for OpenAI API access at https://beta.openai.com/signup/. Once you sign up, you will receive your API key.

2. Choose the type of chatbot you want to create. For example, you can create an FAQ chatbot, a customer support chatbot, or a conversational chatbot.

3. Use OpenAI's GPT-3 language model to generate responses to user input. You can use the API to train the language model on your chatbot's intended use case/s.

4. Use Natural Language Processing (NLP) techniques to understand user input and provide relevant responses. You can use OpenAI's API to extract entities (such as dates and names) from user input.

5. Use Machine Learning to continually improve the chatbot's ability to understand and respond to user input.

6. Integrate the chatbot with your preferred messaging platform or channel (e.g., web chat, social media, etc.) using API connectors.

7. Test your chatbot frequently, and use user feedback to improve its performance and provide the best possible experience for your users.

A. SIMPLE CHATGPT USING OPENAI

CODE:

```
Pip install openai

import openai

openai.api_key = "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"

completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=[{"role": "user", "content": "Give me 3 ideas that i could build using openai apis"}])

print(completion.choices[0].message.content)
```

OUTPUT:

1. Personalized Content Recommendation System: Develop an AI-powered content recommendation system that suggests personalized content to users based on their interests and search history. Use OpenAI's language generation APIs to generate relevant content descriptions and summaries, and employ their natural language processing (NLP) APIs to understand user preferences and interests.

2. Intelligent Chatbot: Build a conversational AI-enabled chatbot that can answer customer queries, provide helpful recommendations, and complete transactions seamlessly. Use OpenAI's language processing APIs to train the chatbot to understand user inputs and respond in natural language. Integration with other APIs such as payment gateways and customer databases can make the chatbot efficient and effective.

3. Fraud Detection System: Develop a machine learning model that can identify and prevent fraudulent activities using OpenAI's anomaly detection and classification APIs. Train the model using historical data of fraudulent transactions, and use the APIs to continuously scan for and identify suspicious activities. Such a system can be deployed in a range of applications such as finance or e-commerce platforms.

B. CHATGPT ASSISTANT USING OPENAI

CODE:

```python
import openai

openai.api_key = "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"

messages = []
system_msg = input("What type of chatbot would you like to create?\n")
messages.append({"role": "system", "content": system_msg})
```

```python
print("Your new assistant is ready! Type your query")

while input != "quit()":

    message = input()

    messages.append({"role": "user", "content": message})

    response = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=messages)

    reply = response["choices"][0]["message"]["content"]

    messages.append({"role": "assistant", "content": reply})

    print("\n" + reply + "\n")
```

OUTPUT:

What type of chatbot would you like to create?

Nila' s personal chatbot

(ctrl enter)

Your new assistant is ready!

C.   CHATBOT CHAT ASSISTANT WEBSITE

CODE:

import openai

import gradio

```python
openai.api_key = "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"

messages = [{"role": "system", "content": "You are a financial experts that specializes in
real estate investment and negotiation"}]

def CustomChatGPT(user_input):

    messages.append({"role": "user", "content": user_input})

    response = openai.ChatCompletion.create(

        model = "gpt-3.5-turbo",

        messages = messages

    )

    ChatGPT_reply = response["choices"][0]["message"]["content"]

    messages.append({"role": "assistant", "content": ChatGPT_reply})

    return ChatGPT_reply

demo = gradio.Interface(fn=CustomChatGPT, inputs = "text", outputs = "text", title =
"INTELLIGENT CHATBOT")

demo.launch(share=True)
```
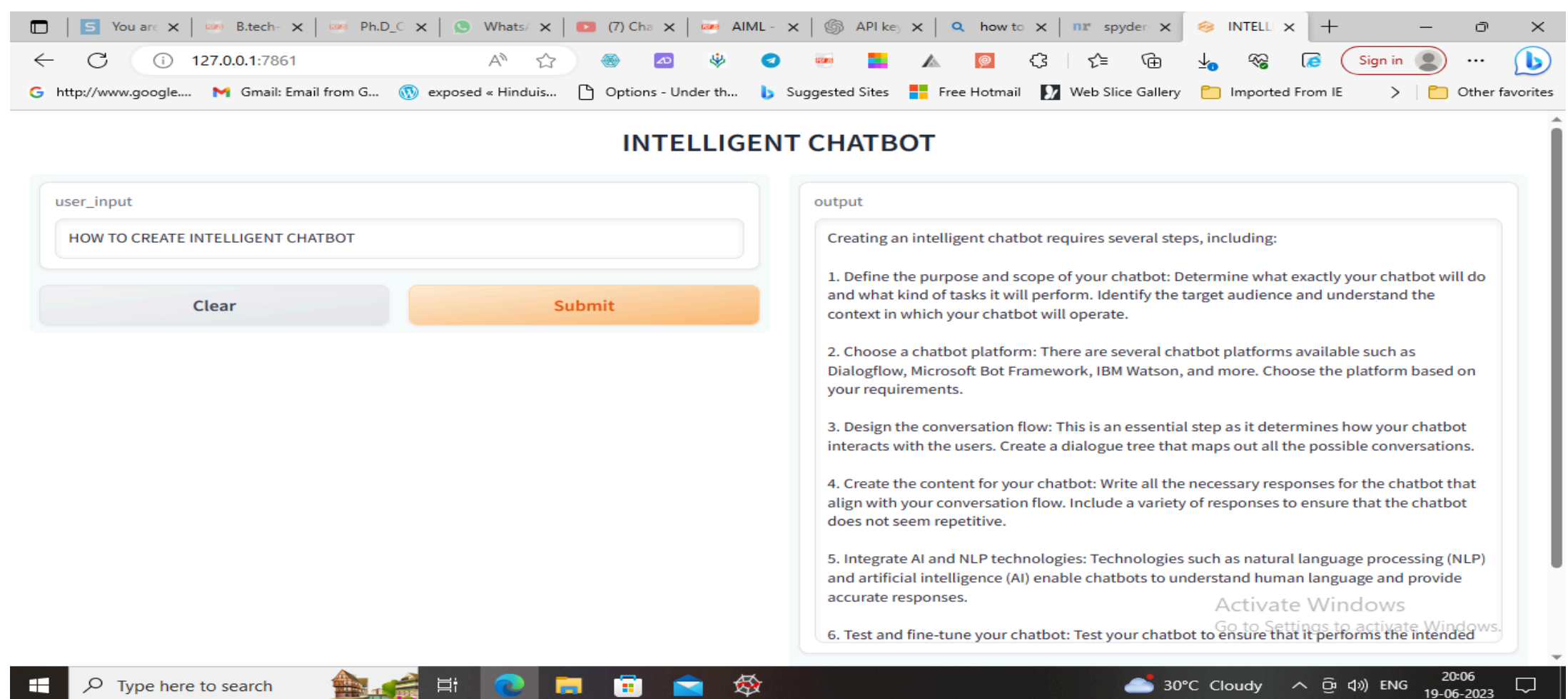
OUTPUT:

INTELLIGENT CHATBOT

user_input

HOW TO CREATE INTELLIGENT CHATBOT

Clear | Submit

output

Creating an intelligent chatbot requires several steps, including:

1. Define the purpose and scope of your chatbot: Determine what exactly your chatbot will do and what kind of tasks it will perform. Identify the target audience and understand the context in which your chatbot will operate.

2. Choose a chatbot platform: There are several chatbot platforms available such as Dialogflow, Microsoft Bot Framework, IBM Watson, and more. Choose the platform based on your requirements.

3. Design the conversation flow: This is an essential step as it determines how your chatbot interacts with the users. Create a dialogue tree that maps out all the possible conversations.

4. Create the content for your chatbot: Write all the necessary responses for the chatbot that align with your conversation flow. Include a variety of responses to ensure that the chatbot does not seem repetitive.

5. Integrate AI and NLP technologies: Technologies such as natural language processing (NLP) and artificial intelligence (AI) enable chatbots to understand human language and provide accurate responses.

6. Test and fine-tune your chatbot: Test your chatbot to ensure that it performs the intended

RESULT:

Thus, to build an intelligent chatbox system with Python and dialogue flow was successfully completed and output was verified.