<p style="text-align:center;">**TASK:9**</p>

<p style="text-align:center;">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.</p>

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:

```
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})

print("Your new assistant is ready! Type your query")
while input != "quit()":
    message = input()
```

```python
    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:

```python
import openai

import gradio

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})
```
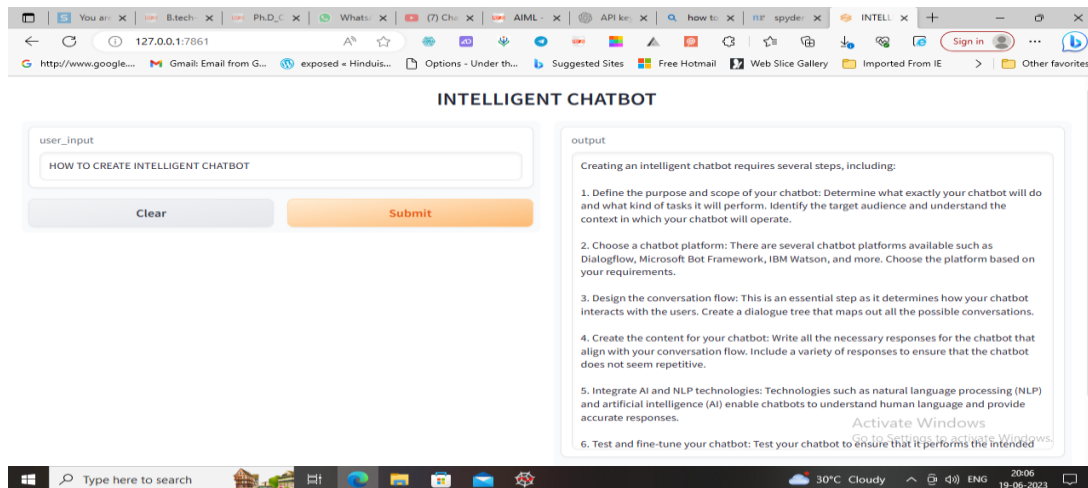
return ChatGPT_reply

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

demo.launch(share=True)

OUTPUT:

Result:

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