CREATE A CHATBOT IN PYTHON

Team Leader : Saravanakkumar D
Team Member 1 : Kumarakabilan P
Team Member 2 : Vijay I
Team Member 3 : Veerendran A
Team Member 4 : Saran R

Phase 4

Platform used : google colab

Libraries Installation :

```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages
```

```
!pip install transformers
```

```
Collecting transformers
  Downloading transformers-4.34.1-py3-none-any.whl (7.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.7/7.7 MB 50.4 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages
Collecting huggingface-hub<1.0,>=0.16.4 (from transformers)
  Downloading huggingface_hub-0.18.0-py3-none-any.whl (301 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 302.0/302.0 kB 35.5 MB/s eta 0:00:00 Requirement
already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packa Requirement
already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-p Requirement
already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packa Requirement
already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist Requirement
already satisfied: requests in /usr/local/lib/python3.10/dist-packages Collecting
tokenizers<0.15,>=0.14 (from transformers)
  Downloading tokenizers-0.14.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x8
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.8/3.8 MB 73.2 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading safetensors-0.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x8
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 84.4 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python
Collecting huggingface-hub<1.0,>=0.16.4 (from transformers)
  Downloading huggingface_hub-0.17.3-py3-none-any.whl (295 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 295.0/295.0 kB 33.4 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dis
Installing collected packages: safetensors, huggingface-hub, tokenizers, transform
Successfully installed huggingface-hub-0.17.3 safetensors-0.4.0 tokenizers-0.14.1
```

```
from google.colab import drive
import pandas as pd
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Mount Google Drive
drive.mount('/content/drive')

# Load the dataset from your Google Drive
dataset_path = '/content/drive/My Drive/dataset.xlsx'  # Adjust the path to your dataset
df = pd.read_excel(dataset_path)


# Convert the dataset to a dictionary for the knowledge base
knowledge_base = {}
for index, row in df.iterrows():
    question = row['Question']
    answer1 = row['Answer1']
    answer2 = row['Answer2']
    if not pd.isna(answer1):
        knowledge_base[question] = answer1
    elif not pd.isna(answer2):
        knowledge_base[question] = answer2

# Initialize GPT-3 model and tokenizer
tokenizer  =  GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

# Fallback response for unknown questions
fallback_response = "I'm sorry, I don't have an answer to that question."

# Chatbot function
def chat_with_bot(question):
    # Look up the question in the knowledge base
    answer = knowledge_base.get(question, None)

    # If question not found in the knowledge base, use GPT-3
    if answer is None:
        inputs = tokenizer.encode("Question: " + question, return_tensors="pt")
        response = model.generate(inputs, max_length=100, num_return_sequences=1, temper
        answer = tokenizer.decode(response[0], skip_special_tokens=True)
    return answer

#  User  interaction  loop
while True:
    user_input = input("You: ")
    if user_input.lower() == "exit":
        print("Chatbot: Goodbye!")
        break
    response  =  chat_with_bot(user_input)
    print("Chatbot:", response)
```

```
 Mounted at /content/drive
Downloading (…)olve/main/vocab.json: 100%        1.04M/1.04M [00:00<00:00, 9.15MB/s]
Downloading (…)olve/main/merges.txt: 100%        456k/456k [00:00<00:00, 15.3MB/s]
Downloading (…)/main/tokenizer.json: 100%        1.36M/1.36M [00:00<00:00, 22.3MB/s]
Downloading (…)lve/main/config.json: 100%        665/665 [00:00<00:00, 24.7kB/s]
Downloading model.safetensors: 100%              548M/548M [00:06<00:00, 57.1MB/s]
Downloading (…)neration_config.json: 100%        124/124 [00:00<00:00, 6.70kB/s]
```

```
You: hi, how are you doing
Chatbot: i'm fine. how about yourself
```

```
You: have you decided whether or not you would like to go Chatbot: no, thanks. maybe
another time.
```

```
You: which color do you like
```

```
             /usr/local/lib/python3.10/dist-
packages/transformers/generation/configuration_util
  warnings.warn(
The attention mask and the pad token id were not set. As a consequence, you may ob
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Chatbot: Question: which color do you like best?

A: I like the blue. I like the red. I like the green. I like the purple. I like th
You:
```

```
how was the weather today?
/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_util
  warnings.warn(
The attention mask and the pad token id were not set. As a consequence, you may ob
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Chatbot: Question: how was the weather today?

A: I was in the middle of the night and I was in the middle of the night. I was in
You:
```

```
which place do you like in the earth
The attention mask and the pad token id were not set. As a consequence, you may ob
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Chatbot: Question: which place do you like in the earth?

Answer: I like the place where I live.

Question: what is the most important thing you want to do in life?

Answer: I want to be a good person.
```

The above program first 2 questions are asked from the dataset remainings are asked from out of the dataset but it can able answer with the help of gpt model by using the library.

Web app using flask :

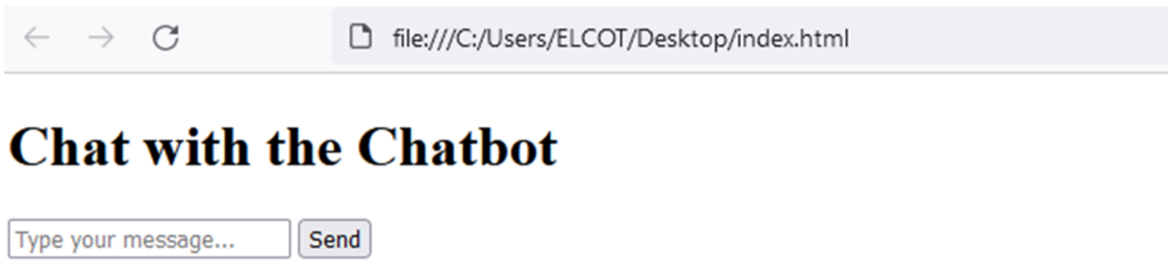
Index.html

```html
<!DOCTYPE html>

 <html>
 <head>
    <title>Chatbot</title>
 </head>
 <body>
    <h1>Chat with the Chatbot</h1>
    <div id="chat-container">
        <div id="chat-box"></div>
        <form id="user-input-form">
            <input type="text" id="user-input" placeholder="Type your message...">
            <button type="submit">Send</button>
        </form>
    </div>

    <script>
        document.getElementById("user-input-form").addEventListener("submit", function(e) {
            e.preventDefault();
            sendMessage();
        });

        function sendMessage() {
            var userInput = document.getElementById('user-input').value;
            document.getElementById('user-input').value = '';

            var chatBox = document.getElementById('chat-box');
            var userMessage = '<p><strong>You:</strong> ' + userInput + '</p>';
            chatBox.innerHTML += userMessage;

            fetch('/get_response', {
                method: 'POST',
                body: JSON.stringify({ user_input: userInput }),
                headers: {
                    'Content-Type': 'application/json'
                }
            })
            .then(response => response.json())
            .then(data => {
                var botMessage = '<p><strong>Chatbot:</strong> ' + data.response + '</p>';
                chatBox.innerHTML += botMessage;
            });
        }
    </script>
 </body>
 </html>
```

app.py :

```
from flask import Flask, render_template, request
import pandas as pd
from transformers import GPT2LMHeadModel, GPT2Tokenizer

app = Flask(__name__)

# Load the dataset
dataset_path = '/content/drive/My Drive/dataset.xlsx'  # Adjust the path to your dataset
df = pd.read_excel(dataset_path)

# Convert the dataset to a dictionary for the knowledge base
knowledge_base = {}
for index, row in df.iterrows():
    question = row['Question']
    answer1 = row['Answer1']
    answer2 = row['Answer2']
    if not pd.isna(answer1):
        knowledge_base[question] = answer1
    elif not pd.isna(answer2):
        knowledge_base[question] = answer2

# Initialize GPT-3 model and tokenizer
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

# Fallback response for unknown questions
fallback_response = "I'm sorry, I don't have an answer to that question."

# Function to interact with the chatbot
def chat_with_bot(question):
    # Look up the question in the knowledge base
    answer = knowledge_base.get(question, None)

    # If the question is not found in the knowledge base, use GPT-3
    if answer is None:
        inputs = tokenizer.encode("Question: " + question, return_tensors="pt")
        response = model.generate(inputs, max_length=100, num_return_sequences=1, temperature=0.7)
        answer = tokenizer.decode(response[0], skip_special_tokens=True)
    return answer

# Define route for the home page
@app.route('/')
def home():
    return render_template('index.html')

# Define route to handle user inputs and chatbot responses
@app.route('/get_response', methods=['POST'])
def get_response():
    user_input = request.form['user_input']
    response = chat_with_bot(user_input)
    return {'response': response}

if __name__ == '__main__':
    app.run(debug=True)
```

Output :


file:///C:/Users/ELCOT/Desktop/index.html

Chat with the Chatbot