CHATGPT JOKEBOT

Building a joke bot using the ChatGPT API and Gradle

In [ ]:
```python
import openai

openai.api_key = "YOUR_API_KEY"
```

- Initialize chatGPT
- Assign the user role and ask a question for testing

In [ ]:
```python
completion = openai.ChatCompletion.create(
    model = "gpt-3.5-turbo",
    messages = [{"role":"user", "content": "What is the capital of Norway?"}]
)

print(completion)
```

```
{
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "message": {
        "content": "The capital of Norway is Oslo.",
        "role": "assistant"
      }
    }
  ],
  "created": 1680030979,
  "id": "chatcmpl-6z95nRP9Hid6JStnXeAZLUyvEbi0q",
  "model": "gpt-3.5-turbo-0301",
  "object": "chat.completion",
  "usage": {
    "completion_tokens": 7,
    "prompt_tokens": 15,
    "total_tokens": 22
  }
}
```

- Extrect the answer to the question, it should be under the choices list as message, role should be assistant and content would be the answer.
- Print it to be sure.

In [ ]:
```python
reply_content = completion.choices[0].message.content
#reply_content2 = completion["choices"][0]["message"]["content"]


print(reply_content)
```

```
The capital of Norway is Oslo.
```

- We need to keep track of the history of questions asked
- We need to ask for user input, this will be important for later

In [ ]:
```
message_history = []

user_input = input(">: ")

print(f"User input is {user_input}")
```

```
<  What is the capital of Nigeria?
User input is What is the capital of Nigeria?
```

- Append the message history and change content to user input instead of writing out the question every time

In [ ]:
```
message_history.append({"role":"user", "content":user_input})
```

- Rewrite the completion variable message_history as input
- Do the same with reply_content

In [ ]:
```
completion = openai.ChatCompletion.create(
    model = "gpt-3.5-turbo",
    messages = message_history,
)

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

- If you want to force chat GPT to give a speciefic answer, you can supply one and append it to message_history

In [ ]:
```
#message_history.append({"role":"assistant", "content":user_input})
```

In [ ]:
```
user_input = input(">: ")

print(f"User input is {user_input}\n")
message_history.append({"role":"user", "content":user_input})

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

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

```
<  What is the capital of Nigeria?
User input is What is the capital of Nigeria?
```

- This is essentially the full program, recurring until you type "exit" to quit the program.

- Use a while loop to keep the program running until you type "exit" to quit the program.

```
In [ ]: message_history = []

        def chat(inp, role = "user"):
            message_history.append({"role": role, "content": inp})

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

            reply_content = completion.choices[0].message.content
            print(reply_content)

            message_history.append({"role": "assistant", "content": reply_content})
            return reply_content

        while True:
        #for i in range():
            user_input = input(">: ")
            if user_input == "exit":
                break
            print(f"The user input is: {user_input}\n")
            chat(user_input)
            print()
```

```
The user input is: What is the capital of Nigeria?

Abuja.
```

- Time to make this more presentable with a GUI
- Import gradio and openai modules
- Supply your api key

```
In [ ]: import gradio as gr
        import openai

        openai.api_key = "YOUR_OPENAI_API_KEY"
```

- Initialize the message history and instruct ChatGPT to be a joke bot
- Force it by setting the initial answer for the bot

```
In [ ]: message_history = [{"role": "user", "content": f"You are a joke bot. I will specify th
                           {"role": "assistant", "content": f"OK"}]
```

- Create a function to compress everything we've done so far to prepare the bot for deployment on gradio

```
In [ ]: def predict(input):
            # tokenize the new input sentence
            message_history.append({"role": "user", "content": f"{input}"})
```

```
    completion = openai.ChatCompletion.create(
      model="gpt-3.5-turbo",
      messages=message_history
    )
    #Just the reply text
    reply_content = completion.choices[0].message.content#.replace('```python', '<pre>

    message_history.append({"role": "assistant", "content": f"{reply_content}"})

    # get pairs of msg["content"] from message history, skipping the pre-prompt:
    response = [(message_history[i]["content"], message_history[i+1]["content"]) for i
    return response
```

- Launch Gradio and assign our function to the text box

In [ ]:
```
# creates a new Blocks app and assigns it to the variable demo.
with gr.Blocks() as demo:

    # creates a new Chatbot instance and assigns it to the variable chatbot.
    chatbot = gr.Chatbot()

    # creates a new Row component, which is a container for other components.
    with gr.Row():
        '''creates a new Textbox component, which is used to collect user input.
        The show_label parameter is set to False to hide the label,
        and the placeholder parameter is set'''
        txt = gr.Textbox(show_label=False, placeholder="Enter text and press enter").s
    '''
    sets the submit action of the Textbox to the predict function,
    which takes the input from the Textbox, the chatbot instance,
    and the state instance as arguments.
    This function processes the input and generates a response from the chatbot,
    which is displayed in the output area.'''
    txt.submit(predict, txt, chatbot) # submit(function, input, output)
    '''
    sets the submit action of the Textbox to a JavaScript function that returns an emp
    This line is equivalent to the commented out line above, but uses a different impl
    The _js parameter is used to pass a JavaScript function to the submit method.'''
    txt.submit(None, None, txt, _js="() => {''}")

demo.launch(share=True)
```

```
Running on local URL:  http://127.0.0.1:7866

Could not create share link. Please check your internet connection or our status pag
e: https://status.gradio.app
```

Chatbot

Enter text and press enter

Use via API · Built with Gradio

```
Out[ ]:
```