

CREATE A CHATBOT USING PYTHON

README FILE:

1. Introduction

This AI chatbot is designed to have text-based conversations with users. It uses a pre-trained model and natural language processing techniques to generate responses to user input.

2. Prerequisites

Before running the code, make sure you have the following dependencies installed:

Python 3.x

Pip (Python package manager)

Required Python libraries (you can install these using pip):

transformers

torch (PyTorch)

INSTALLATION:

To set up the chatbot, follow these steps:

1. Click on the Run button to Install the Packages



```
!pip install langchain
!pip install openai
!pip install gradio
!pip install huggingface_hub
```


2. Click on the Run button to import the required things to build the application



```
import os
import gradio as gr
from langchain.chat_models import ChatOpenAI
from langchain import LLMChain, PromptTemplate
from langchain.memory import ConversationSummaryMemory
```

- **Get the OpenAI API Key and set it as environmental variable**

- Generate API Key



```
OPENAI_API_KEY="OPENAI_API_KEY"|  
os.environ["OPENAI_API_KEY"] = OPENAI_API_KEY
```

- - Go to openai
 - Create a new Secret Key
 - Copy the Secret Key for your use.
- Replace your Open AI API Key with your own API Key
Click on the Run button

- **Assigning the values for template, prompt, and memory**

- You can update the first line of the template provided

Click on the Run button



```
template = """You are a helpful assistant to answer user queries.  
{chat_history}  
User: {user_message}  
Chatbot: """  
  
prompt = PromptTemplate(  
    input_variables=["chat_history", "user_message"], template=template  
)  
  
memory = ConversationBufferMemory(memory_key="chat_history")
```

- **Initializing LLM Chain using Open AI**

- Using ChatOpenAI we are creating an [LLMChain](#)

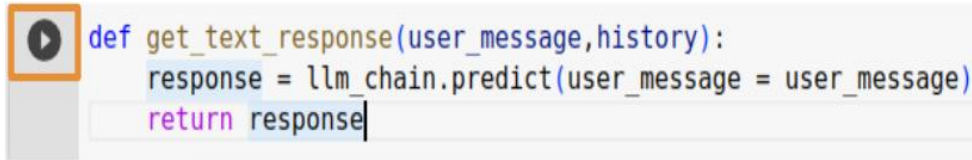
Click on the Run button



```
llm_chain = LLMChain(  
    llm=ChatOpenAI(temperature='0.5', model_name="gpt-3.5-turbo"),  
    prompt=prompt,  
    verbose=True,  
    memory=memory,  
)
```

- **Define a function to generate the response for the question you ask:**

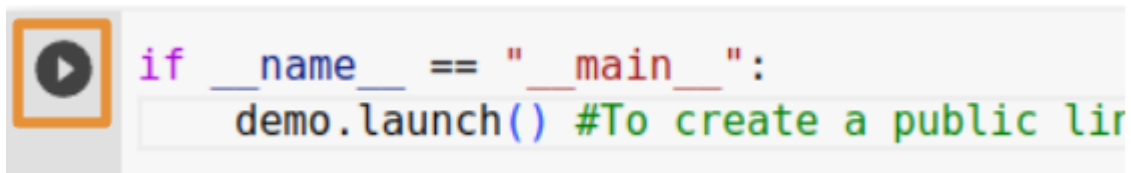
- From the initialized llm_chain we will predict the response.
- Click on the Run button



```
def get_text_response(user_message, history):  
    response = llm_chain.predict(user_message = user_message)  
    return response
```

- **Launch your ChatBot with Gradio APP**

- Click on the Run button to launch the App



```
if __name__ == "__main__":  
    demo.launch() #To create a public link
```

-

Now you can try asking questions in your ChatBot

To identify the error you are getting please add **debug=True** while launching the gradio app.

```
if __name__ == "__main__":  
    demo.launch(debug=True)  
  
def get_text_response(user_message, history):  
    try:  
        response = llm_chain.predict(user_message = user_message)  
    except Exception as e:  
        print("Error:", e)  
    try:  
        print("Error:", e.error.message)  
        response = "Failed to reply: " + e.error.message  
    except Exception as e:  
        response = "Failed to reply"  
  
    return response
```

4.CUSTOMIZATION:

You can customize the chatbot's behavior by modifying the configuration files and fine-tuning the model. Here are some points to consider:

Model Configuration: You can modify the chatbot's behavior by changing the model's hyperparameters, such as the model size, learning rate, and training data.

Data: You can improve the chatbot's performance by training it on domain-specific data or fine-tuning it with additional datasets.

Integration: You can integrate this chatbot with other applications or platforms to make it available to a wider audience.

5.CONTRIBUTING:

We welcome contributions to improve this chatbot. If you would like to contribute, please follow these guidelines:

Fork the repository.

Create a new branch for your feature or bug fix.

Make your changes and test them.

Submit a pull request to the main repository's main branch.

6.CONCLUSION:

In conclusion, generative AI chatbots represent a significant advancement in natural language processing and human-computer interaction. These chatbots have the ability to understand and generate human-like text, enabling more engaging and personalized conversations. They have found applications in customer support, content creation, and various industries, streamlining processes and improving user experiences. However, challenges remain, including the need to mitigate biases and improve the quality of responses. As these technologies continue to evolve, the potential for generative AI chatbots to revolutionize communication and information exchange is substantial, and their impact on various aspects of our lives is expected to grow significantly in the coming years. Ethical considerations and ongoing research will be crucial in ensuring their responsible and effective use.