ChatBot AI project | CODESOFT Internship - submitted by Harshad Dhokane

---

Installs several Python libraries using pip:

---

fastapi: A modern, fast, (high-performance) web framework for building APIs with Python 3.7+ based on standard Python type hints.

uvicorn: An ASGI server for serving FastAPI applications.

nest-asyncio: A library to allow nested use of asyncio event loops, useful for running asynchronous code in environments that use asyncio.

pyngrok: A tool to create secure tunnels to localhost, which is useful for exposing your local server to the internet.

transformers: A library by Hugging Face for working with state-of-the-art natural language processing models.

```
# Install necessary libraries
!pip install fastapi uvicorn nest-asyncio pyngrok transformers
```

```
Collecting fastapi
  Downloading fastapi-0.112.2-py3-none-any.whl.metadata (27 kB)
Collecting uvicorn
  Downloading uvicorn-0.30.6-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.10/dist-packages (1.6.0)
Collecting pyngrok
  Downloading pyngrok-7.2.0-py3-none-any.whl.metadata (7.4 kB)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.42.4)
Collecting starlette<0.39.0,>=0.37.2 (from fastapi)
  Downloading starlette-0.38.2-py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from fastapi) (4.12.2)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.10/dist-packages (from uvicorn) (8.1.7)
Collecting h11>=0.8 (from uvicorn)
  Downloading h11-0.14.0-py3-none-any.whl.metadata (8.2 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.10/dist-packages (from pyngrok) (6.0.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.15.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.2
Requirement already satisfied: numpy<2.0,>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.5.15)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.4)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.5)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.2->t
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8.1,!=
Requirement already satisfied: pydantic-core==2.20.1 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8.1,!=2
Requirement already satisfied: anyio<5,>=3.4.0 in /usr/local/lib/python3.10/dist-packages (from starlette<0.39.0,>=0.37.2->fast
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (202
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.4.0->starlette<0.39.0,
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.4.0->starlette<0.39.
Downloading fastapi-0.112.2-py3-none-any.whl (93 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 93.5/93.5 kB 6.0 MB/s eta 0:00:00
Downloading uvicorn-0.30.6-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 kB 4.0 MB/s eta 0:00:00
Downloading pyngrok-7.2.0-py3-none-any.whl (22 kB)
Downloading h11-0.14.0-py3-none-any.whl (58 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 3.2 MB/s eta 0:00:00
Downloading starlette-0.38.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 3.9 MB/s eta 0:00:00
Installing collected packages: pyngrok, h11, uvicorn, starlette, fastapi
Successfully installed fastapi-0.112.2 h11-0.14.0 pyngrok-7.2.0 starlette-0.38.2 uvicorn-0.30.6
```

Import necessary modules and libraries:

---

nest_asyncio: Adjusts the asyncio event loop to work within Jupyter notebooks, enabling asynchronous code execution.

pyngrok: Provides functionality to expose the local FastAPI server to the internet through secure tunnels.

fastapi: The web framework for building the API.

pydantic: A data validation library for defining and validating data models (using BaseModel).

transformers: A library by Hugging Face for working with transformer models (e.g., GPT-2).

uvicorn: An ASGI server for running FastAPI applications.

The nest_asyncio.apply() call ensures that the asyncio event loop can be used in a Jupyter notebook environment.

```
# Import required modules
import nest_asyncio
from pyngrok import ngrok
from fastapi import FastAPI
from pydantic import BaseModel
from transformers import GPT2LMHeadModel, GPT2Tokenizer
import uvicorn

# Allow the event loop to run in Jupyter notebooks
nest_asyncio.apply()
```

Now lets configures the FastAPI application with CORS support and an ngrok tunnel:

CORSMiddleware from fastapi.middleware.cors: Adds Cross-Origin Resource Sharing (CORS) support to the FastAPI app, allowing it to handle requests from different origins. In this case, it is configured to allow all origins, HTTP methods, and headers.

ngrok.set_auth_token(): Configures ngrok with your authentication token for secure tunneling. Make sure to replace the placeholder token with your actual token.

ngrok.connect(8000): Starts an ngrok tunnel to expose the local FastAPI server running on port 8000 to the internet. It returns a public URL which is printed out.

app = FastAPI(): Initializes a FastAPI application instance.

The app.add_middleware() function configures CORS settings, allowing all origins, credentials, methods, and headers to interact with the FastAPI app.

```
from fastapi.middleware.cors import CORSMiddleware
# Set up ngrok with your authentication token
ngrok.set_auth_token("2lKjRdepOWUEK4EnQcofmY7ej7y_66CQJ7tocTqdMRW2xhivw")  # Replace with your actual authtoken

# Start the ngrok tunnel
public_url = ngrok.connect(8000)
print(f"Public URL: {public_url}")

# Initialize the FastAPI app
app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Allow all origins, or specify the allowed origins
    allow_credentials=True,
    allow_methods=["*"],  # Allow all HTTP methods
    allow_headers=["*"],  # Allow all headers
)
```

```
Public URL: NgrokTunnel: "https://ab7a-35-197-125-60.ngrok-free.app" -> "http://localhost:8000"
```

Now lets loads a pre-trained model and tokenizer:

---

model_name = "microsoft/DialoGPT-large": Specifies the model name. In this case, it’s the microsoft/DialoGPT-large model, which is a variant of GPT-2 fine-tuned for conversational responses.

GPT2Tokenizer.from_pretrained(model_name): Loads the tokenizer associated with the specified model. The tokenizer converts text into tokens that the model can understand.

GPT2LMHeadModel.from_pretrained(model_name): Loads the pre-trained model itself. This model is used to generate responses based on input tokens.

```
# Load the pre-trained model and tokenizer
model_name = "microsoft/DialoGPT-large"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json:  100%                                614/614  [00:00<00:00,  15.2kB/s]
vocab.json:  100%                                   1.04M/1.04M  [00:00<00:00,  9.07MB/s]
merges.txt:  100%                                   456k/456k  [00:00<00:00,  10.3MB/s]
config.json:  100%                                  642/642  [00:00<00:00,  26.3kB/s]
pytorch_model.bin:  100%                                1.75G/1.75G  [00:15<00:00,  217MB/s]
WARNING:pyngrok.process.ngrok:t=2024-08-29T16:19:47+0000 lvl=warn msg="failed to open private leg" id=a31b0fe8bd77 privaddr=loca
generation_config.json:  100%                              124/124  [00:00<00:00,  6.27kB/s]
```

Lets defines data model and a function for generating responses from the conversational model:

Define a Data Model for the Request:

---

class ChatRequest(BaseModel): Defines a data model named ChatRequest using pydantic.BaseModel.

message: str: Specifies a single field message of type str, which will be used to receive chat messages in the request.

Function to Generate a Response from the Model:

generate_response(input_text, model, tokenizer, max_length=1000): Defines a function to generate a response using the pre-trained model.

input_ids = tokenizer.encode(input_text + tokenizer.eos_token, return_tensors='pt'): Encodes the input text into tokens and appends the end-of-sequence token. Converts it to a tensor suitable for the model.

response_ids = model.generate(...): Uses the model to generate a sequence of tokens as a response.

Parameters:

max_length=1000: Specifies the maximum length of the response.

pad_token_id=tokenizer.eos_token_id: Pads the sequence to the end-of-sequence token ID.

top_p=0.95 and top_k=50: Parameters for controlling the randomness and diversity of the generated text.

response_text = tokenizer.decode(response_ids[:, input_ids.shape[-1]:][0], skip_special_tokens=True): Decodes the generated token IDs back into text, skipping any special tokens.

return response_text: Returns the generated response text.

```
# Define a data model for the request
class ChatRequest(BaseModel):
    message: str

# Function to generate a response from the model
def generate_response(input_text, model, tokenizer, max_length=1000):
    input_ids = tokenizer.encode(input_text + tokenizer.eos_token, return_tensors='pt')
    response_ids = model.generate(
        input_ids,
        max_length=max_length,
        pad_token_id=tokenizer.eos_token_id,
        top_p=0.95,
        top_k=50
    )
    response_text = tokenizer.decode(response_ids[:, input_ids.shape[-1]:][0], skip_special_tokens=True)
    return response_text
```

Now lets set up the /chat endpoint for the FastAPI application and starts the server:

Define the /chat Endpoint:

@app.post("/chat"): Defines a POST endpoint at the path /chat.

async def chat_endpoint(request: ChatRequest): Asynchronous function to handle POST requests. It expects a ChatRequest object containing the user message.

user_message = request.message: Extracts the user message from the request.

response_text = generate_response(user_message, model, tokenizer): Calls the generate_response function to get a response from the model.

return {"response": response_text}: Returns a JSON response with the generated text.

Run the FastAPI App:

uvicorn.run(app, host="0.0.0.0", port=8000): Starts the FastAPI application on 0.0.0.0 (all network interfaces) and port 8000. This makes the app accessible from outside the local machine, suitable for testing with ngrok.

```
# Define the /chat endpoint
@app.post("/chat")
async def chat_endpoint(request: ChatRequest):
    user_message = request.message
    response_text = generate_response(user_message, model, tokenizer)
    return {"response": response_text}

# Run the FastAPI app
uvicorn.run(app, host="0.0.0.0", port=8000)
```

```
INFO:     Started server process [790]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "OPTIONS /chat HTTP/1.1" 200 OK
/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_utils.py:545: UserWarning: `do_sample` is set to `
  warnings.warn(
The attention mask is not set and cannot be inferred from input because pad token is same as eos token.As a consequence, you may
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "OPTIONS /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
INFO:     2402:8100:3189:cc7f:ec33:17af:babf:3062:0 - "POST /chat HTTP/1.1" 200 OK
```

Start coding or generate with AI.