QUESTION – ANSWERING WITH BERT

The Question – Answering system is created using pretrained BERT model. And an API is created with FastAPI and built using docker container.

1)Set up the Question Answering System:

Install the required dependencies:

```
pip install transformers, fastapi, uvicorn
```

2)Create a Python file(tap.py) and import the necessary libraries:

```
from fastapi import FastAPI

from pydantic import BaseModel

from transformers import BertForQuestionAnswering, BertTokenizer, pipeline
```

3)Load the BERT question answering model and tokenizer:

```
model_name = 'bert-large-uncased-whole-word-masking-finetuned-squad'

model = BertForQuestionAnswering.from_pretrained(model_name)

tokenizer = BertTokenizer.from_pretrained(model_name)

nlp = pipeline("question-answering", model=model, tokenizer=tokenizer)
```

4)Define the FastAPI app and create a data model:

```
app = FastAPI()

class QuestionContext(BaseModel):

    question: str

    context: str
```

5)Create an endpoint to handle the prediction:

```
@app.post('/predict')
def predict_answer(question_context: QuestionContext):
    answer = nlp({'question': question_context.question, 'context': question_context.context})
    return {'answer': answer['answer']}
```

6)Create a Dockerfile:

Create a file called Dockerfile (without any file extension) in the same directory as tapp.py.

Open the Dockerfile and add the below content:

```
FROM python:3.8-slim-buster

WORKDIR /tapp

COPY requirements.txt .

RUN pip install torch==1.9.0

RUN pip install --no-cache-dir -r requirements.txt

COPY . /tapp

EXPOSE 8000

CMD ["uvicorn", "tapp:app", "--host", "0.0.0.0", "--port", "8000"]
```

7)Create a requirements.txt file:

Create a file called requirements.txt in the same directory as tapp.py and add the following :

```
transformers==4.11.3

fastapi==0.68.0

uvicorn==0.15.0
```

8)Build the Docker image:

Open a terminal or command prompt and navigate to the directory containing the Dockerfile, tapp.py, and requirements.txt files.

Build the Docker image using the following command:

```
docker build -t myy-appi .
```

9)Run the Docker container:

Once the Docker image is built, run a container based on that image:

```
docker run -d -p 8000:8000 myy-appi
```

The container should now be running, and the API will be accessible at http://localhost:8000/predict.

10)Test the API:

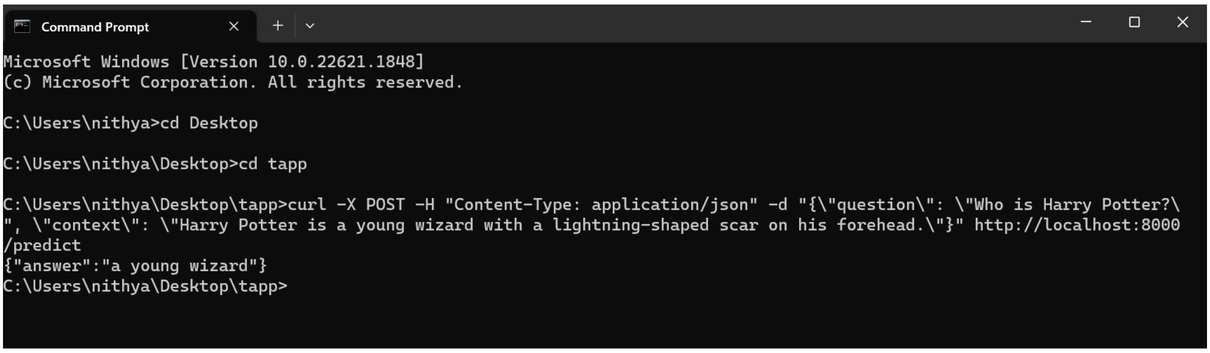
Test the API by using localhost:8000/docs,or

Use an API testing tool like cURL, to send POST requests to http://localhost:8000/predict with JSON payloads containing the question and context.

For example, using cURL, run the following command in command prompt to make a prediction:

```
curl -X POST -H "Content-Type: application/json" -d "{\"question\": \"Who is Harry Potter?\", \"context\": \"Harry Potter is a young wizard with a lightning-shaped scar on his forehead.\"}" http://localhost:8000/predict
```

OUTPUT:



```
Microsoft Windows [Version 10.0.22621.1848]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nithya>cd Desktop

C:\Users\nithya\Desktop>cd tapp

C:\Users\nithya\Desktop\tapp>curl -X POST -H "Content-Type: application/json" -d "{\"question\": \"Who is Harry Potter?\
", \"context\": \"Harry Potter is a young wizard with a lightning-shaped scar on his forehead.\"}" http://localhost:8000
/predict
{"answer":"a young wizard"}
C:\Users\nithya\Desktop\tapp>
```

To use the files in GitHub, clone the GitHub repository containing files to your local machine.

Make sure you have Docker installed on your machine. If you don't have it, download and install Docker from the official website:

Build the Docker image:

Open a terminal or command prompt and navigate to the directory where you cloned the repository.

Build the Docker image using the following command:

```
docker build -t myy-appi .
```

Run the Docker container:

Once the Docker image is built, you can run a container based on that image:

```
docker run -d -p 8000:8000 myy-appi
```

The container should now be running, and the API will be accessible at http://localhost:8000/predict.

Test the API:

Test the API by using localhost:8000/docs,or

You can use an API testing tool like cURL, to send POST requests to http://localhost:8000/predict with JSON payloads containing the question and context.

For example, using cURL, you can run the following command to make a prediction:

```
curl -X POST -H "Content-Type: application/json" -d "{\"question\": \"Who is Harry Potter?\", \"context\": \"Harry Potter is a young wizard with a lightning-shaped scar on his forehead.\"}" http://localhost:8000/predict
```

The response will contain the predicted answer to the question.