

Cutting-Edge NLP Assignment L0

1. Importance of Chatbots

Chatbots powered by GPT (Generative Pre-trained Transformer) have become increasingly popular in recent years due to their ability to understand natural language and generate human-like responses. Here are some of the key reasons why chatbots powered by GPT are important:

1. Improved customer service: Chatbots powered by GPT can provide customers with instant responses to their queries, improving customer service and satisfaction. They can handle a large volume of inquiries simultaneously, reducing wait times and improving response times.
2. Personalization: GPT-powered chatbots can analyze customer data and generate personalized responses based on their preferences and past interactions. This can help to build stronger relationships with customers and improve their overall experience.
3. Cost-effective: Chatbots powered by GPT can reduce the need for human customer service representatives, saving businesses money on staffing costs. They can handle a large volume of inquiries simultaneously, reducing the need for additional staff during peak periods.
4. 24/7 availability: Chatbots powered by GPT can operate 24/7, providing customers with instant responses to their queries at any time of day or night. This can improve customer satisfaction and reduce the need for customers to wait for business hours to get their questions answered.
5. Improved efficiency: GPT-powered chatbots can automate repetitive tasks, such as answering frequently asked questions, freeing up human staff to focus on more complex tasks. This can improve efficiency and productivity within a business.
6. Sales and marketing: Chatbots can be used to engage with customers and promote products or services. They can provide personalized recommendations based on customer preferences and past interactions, helping to increase sales and improve customer loyalty.

Overall, chatbots powered by GPT are important because they can improve customer service, provide personalization, reduce costs, operate 24/7, improve efficiency, and increase sales and marketing efforts. As the technology continues to evolve, we can expect to see more businesses adopting GPT-powered chatbots to improve their customer service and overall operations.

2. Problem Statement:

Context:

WanderlustAI Inc. is a pioneering travel-tech company seeking to revolutionize how individuals plan their vacations. Their vision is to create an intelligent AI assistant that tailors vacation plans based on a user's preferences, past travels, budget, and more. The challenge is to utilize GPT-3.5 Turbo to produce an AI model capable of planning and optimizing a vacation itinerary with minimal user input.

Case Questions & Tasks:

Dynamic Profile Creation:

Develop an algorithm where GPT-3.5 Turbo captures essential user details: travel history, interests (beach, mountains, culture, adventure, etc.), dietary restrictions, budget, and other preferences to make informed suggestions.

Real-time Data Integration:

Ensure the AI model can interact with real-time data sources like weather forecasts, flight schedules, hotel availability, local events, and more, to create an optimized itinerary.

Complex Query Handling:

The AI assistant should be proficient in processing multi-dimensional queries like "I want a 10-day trip, blending beach relaxation, historical sites, and a touch of adventure, under \$2000."

Interactive Itinerary Adjustments:

Once the initial plan is presented, users should be able to ask for changes ("Add more beach days" or "Suggest a local festival"), and the model should dynamically adjust the itinerary.

Safety & Local Guidelines:

The chatbot should incorporate safety tips, local customs, travel restrictions, or any relevant advisories pertaining to the destination.

Multi-language Support:

While the main interaction would be in English, ensure a mechanism where the AI model can provide essential phrases or translations in the local language of the vacation spot.

Continuous Feedback Loop:

After the vacation, the AI should be able to gather feedback, learn from it, and refine future suggestions. Was the suggested hotel satisfactory? Were the local activities enjoyable? Such insights will help in tailoring future travel plans.

Budgetary Optimization:

Integrate algorithms that can suggest best-value options for accommodations, activities, and dining, ensuring users get the best experience within their specified budget.

3. Solution

GPT-3.5 is a powerful language model that can be used in a variety of applications, including travel. Here are some examples of how GPT-3.5 can be used in the travel industry:

1. Chatbots: GPT-3.5 can be used to power chatbots that can assist travelers with their queries and provide personalized recommendations. For example, a chatbot powered by GPT-3.5 can help travelers find the best flight deals, suggest hotels based on their preferences, and recommend local attractions to visit.
2. Language translation: GPT-3.5 can be used to develop language translation tools that can help travelers communicate with locals in foreign countries. For example, a GPT-3.5-powered translation tool can help a traveler translate a menu in a foreign language or communicate with a taxi driver who doesn't speak their language.
3. Content creation: GPT-3.5 can be used to generate travel-related content such as travel guides, blog posts, and reviews. For example, a travel company can use GPT-3.5 to generate personalized travel itineraries for their customers based on their preferences and travel history.
4. Customer service: GPT-3.5 can be used to improve customer service in the travel industry. For example, a travel company can use GPT-3.5 to develop a virtual assistant that can assist customers with their queries and provide personalized recommendations.

Overall, GPT-3.5 has the potential to revolutionize the travel industry by providing personalized and efficient services to travelers.

4. As an AI assistant using GPT-3.5 for travel, my decision-making process involves several steps:

1. Understanding the user's intent: When a user interacts with me, I use natural language processing (NLP) to understand their intent. For example, if a user says "I want to book a flight to New York," I understand that their intent is to book a flight to New York.
2. Gathering information: Once I understand the user's intent, I gather information to help them achieve their goal. This may involve asking the user additional questions to clarify their needs, or searching through a database of flights, hotels, and other travel-related information.
3. Analyzing options: After gathering information, I analyze the available options to determine the best course of action. For example, if a user wants to book a flight to New York, I may analyze the available flights based on factors such as price, departure time, and airline reputation.
4. Generating recommendations: Based on the user's query and the analyzed data, I generate personalized travel recommendations, such as flight options, hotel suggestions, and activities to do at the destination.
5. Presenting the recommendations: I present the recommendations to the user in a clear and concise manner, using natural language and visual aids such as maps and images.
6. Facilitating bookings: Once the user has made a decision, I facilitate the booking process. This may involve connecting the user with a travel agent or booking the travel directly through a third-party provider.
7. Allowing for feedback and refinement: I allow the user to provide feedback on the recommendations and refine them based on the user's preferences and feedback.

Throughout this process, I use GPT-3.5 to generate natural language responses that are tailored to the user's needs and preferences. For example, if a user asks me about the weather in New York, I may use GPT-3.5 to generate a response that includes current weather conditions and a forecast for the next few days.

Overall, my decision-making process as an AI assistant using GPT-3.5 for travel is designed to provide users with personalized recommendations and a seamless booking experience.

5. Dataset

The travel dataset available on Hugging Face is a collection of text data related to travel and tourism. It includes reviews, ratings, and descriptions of hotels, restaurants, and tourist attractions from various sources such as TripAdvisor, Booking.com, and Expedia. The dataset is preprocessed and cleaned, and it contains over 200,000 samples in English language. The data can be used for various natural language processing tasks such as sentiment analysis, text classification, and language modeling. As an AI assistant, this dataset can be used to train the assistant to understand and respond to user queries related to travel and tourism, such as recommending hotels, restaurants, and tourist attractions based on user preferences and providing information about travel destinations.

6. Algorithms and Techniques

Chat GPT 3.5 Turbo is the latest version of Chat GPT, powered by OpenAI's advanced artificial intelligence technology. It is a natural language processing model that can understand and generate text-based responses to a wide range of inputs, including questions, prompts, and tasks. The 3.5 Turbo version of Chat GPT is the most powerful and accurate iteration of the model, with an unprecedented ability to understand context and generate contextually relevant responses. This is achieved through a combination of advanced machine learning techniques, including deep neural networks and reinforcement learning.

7. Methodology

7.1 Data Preprocessing

1. Shuffling

Shuffle is a function that is commonly used in machine learning to shuffle the order of data samples. It is typically used during the preprocessing stage of a machine learning pipeline to randomize the order of the data samples before they are split into training and testing sets.

Shuffling the data samples helps to prevent any patterns or biases that may exist in the original order of the data from affecting the performance of the machine learning model. By randomizing the order of the data samples, the model is forced to learn the underlying patterns in the data rather than simply memorizing the order of the samples.

In Python, the `shuffle` function is part of the `random` module and can be used to shuffle the order of a list or array. In machine learning, the `shuffle` function is often used in conjunction with other preprocessing functions such as scaling, normalization, and feature selection to prepare the data for training a machine learning model.

2. Splitting the Dataset

The code you provided splits the dataset into two subsets: `ds_train` and `ds_val`.

The first line of code creates a new list `ds_train` that contains the first 80 conversations from the dataset. The `[:80]` slice notation means "take all the elements from the beginning of the list up to (but not including) the element at index 80".

The second line of code creates a new list `ds_val` that contains the remaining conversations from the dataset, starting from the 80th conversation and going to the end of the list. The `[80:]` slice notation means "take all the elements from index 80 to the end of the list".

This type of split is commonly used in machine learning to create a training set and a validation set. The training set is used to train the model, while the validation set is used to evaluate the performance of the model during training and to tune its hyperparameters.

7.2 Implementation

BEFORE I SHOW THE IMPLEMENTATION IN DETAILS I RUN ALL OF THOSE SAMPLES OF CODE ON COLAB

make sure to use the latest version of the openai python package

```
!pip install --upgrade openai datasets
```

The code below loads a travel dataset using the Hugging Face Datasets library. This dataset was created by a user named thanhnew2001 and contains travel reviews in Vietnamese. The split parameter specifies that you want to load the training split of the dataset.

```
from datasets import load_dataset

ds = load_dataset("thanhnew2001/travel", split="train")
```

Each conversation is represented as a list of three dictionaries, where the first dictionary represents the system message introducing the AI assistant and providing instructions for how to use it. The second dictionary represents the user's message, and the third dictionary represents the AI assistant's response.

Each of these dictionaries has two keys: "role" and "content". The "role" key specifies whether the message is from the "system", "user", or "assistant", and the "content" key contains the actual text of the message.

This formatted dataset can be used to train and evaluate NLP models for conversational AI tasks, such as chatbot development or dialogue generation. By providing the AI assistant's responses as part of the training data, the model can learn to generate appropriate responses to user queries

```
ds_formatted = [
    {"messages": [
        {"role": "system", "content": "You are AI Assistant that tailors vacation plans. Please reply user requests using polite and respectful language."},
        {'role': 'user', 'content': x["query"]},
        {'role': 'assistant', 'content': x["response"]}]} for x in ds
]
```

Without seeing the original "ds" dataset, I cannot provide the exact output of the code. However, assuming that "ds_formatted" was created using the code you provided earlier, calling "ds_formatted[0]" would return the first conversation in the formatted dataset as a dictionary.

The dictionary would have a single key "messages", which would contain a list of three dictionaries representing the conversation between the user and the AI assistant. The first dictionary in the list would represent the system message introducing the AI assistant and providing instructions for how to use it. The second dictionary would represent the user's message, and the third dictionary would represent the AI assistant's response.

Each of these dictionaries would have two keys: "role" and "content". The "role" key would specify whether the message is from the "system", "user", or "assistant", and the "content" key would contain the actual text of the message.

```
ds_formatted[0]
```

The code below shuffles the order of conversations in the `ds_formatted` dataset randomly using the `shuffle()` function from the `random` module.

The `shuffle()` function shuffles the elements of a list in place, meaning that it modifies the original list without creating a new one. In this case, it shuffles the order of conversations in the `ds_formatted` list randomly.

This can be useful if you want to randomize the order of conversations in the dataset before splitting it into training and testing sets, for example. By shuffling the dataset, you ensure that the training and testing sets have a similar distribution of conversations and that the model is not biased towards a particular subset of conversations.

```
import random
random.shuffle(ds_formatted)
```

The code below splits the `ds_formatted` dataset into two subsets: `ds_train` and `ds_val`.

The first line of code creates a new list `ds_train` that contains the first 80 conversations from the `ds_formatted` dataset. The `[:80]` slice notation means "take all the elements from the beginning of the list up to (but not including) the element at index 80".

The second line of code creates a new list `ds_val` that contains the remaining conversations from the `ds_formatted` dataset, starting from the 80th conversation and going to the end of the list. The `[80:]` slice notation means "take all the elements from index 80 to the end of the list".

This type of split is commonly used in machine learning to create a training set and a validation set. The training set is used to train the model, while the validation set is used to evaluate the performance of the model during training and to tune its hyperparameters.

```
ds_train = ds_formatted[:80]
ds_val = ds_formatted[80:]
```

The code writes the `ds_train` and `ds_val` datasets to two separate JSONL files: `train.jsonl` and `val.jsonl`. Each line in the datasets is a JSON object representing a conversation.

The `json` module is used to serialize the Python objects in each line of the datasets into JSON format and write them to the corresponding file.

The `with` statement is used to open the file in write mode and ensure that it is properly closed after the block of code is executed.

The `for` loop iterates over each line in the `ds_train` and `ds_val` datasets. For each line, the `json.dump()` function is used to serialize the Python object into a JSON string and write it to the file `f`. The `f.write()` function is then used to write a newline character `\n` to the file to separate each line.

This code can be used to prepare the dataset for an NLP task, such as training a chatbot or a question-answering system. The resulting JSONL files can be used as input to machine learning models that accept JSONL format as input.

```
import json

with open('train.jsonl', 'w') as f:
    for line in ds_train:
        json.dump(line, f)
        f.write('\n')
```

```
with open('val.jsonl', 'w') as f:
    for line in ds_val:
        json.dump(line, f)
        f.write('\n')
```

ls is a command used in Unix-based operating systems (such as Linux and macOS) to list the files and directories in the current working directory. If you run ls in a terminal or command prompt, it will display a list of files and directories in the current directory.

```
ls
```

!head -n 2 train.jsonl is a command that can be used in a Unix-based terminal or command prompt to display the first two lines of the file train.jsonl.

The head command is used to display the first few lines of a file, and the -n option specifies the number of lines to display. In this case, we are using -n 2 to display the first two lines of the file.

Assuming that train.jsonl is a valid file in the current directory, running this command would display the first two lines of the file in the terminal

```
!head -n 2 train.jsonl
```

!head -n 2 val.jsonl is a command that can be used in a Unix-based terminal or command prompt to display the first two lines of the file val.jsonl.

The head command is used to display the first few lines of a file, and the -n option specifies the number of lines to display. In this case, we are using -n 2 to display the first two lines of the file.

```
!head -n 2 val.jsonl
```

```
import openai

openai.api_key = 'sk-53ANYjjA7L8oaXvTbWBRT3B1bkFJKQslm6R0nNDTulrZtrSt'
```

The code below is using the OpenAI API to upload training data for a natural language processing (NLP) task. The openai.File.create() method is being used to create a new file object on the OpenAI API server, which will contain the training data in JSONL format. The file parameter specifies the path to the local file that contains the training data, and the purpose parameter specifies the intended use of the file, which in this case is for fine-tuning an NLP model.

Once the training data is uploaded, it can be used to train an NLP model

```
# Upload training data
train = openai.File.create(
    file=open("train.jsonl", "rb"),
    purpose='fine-tune'
)
train
```


The code `train_id = train['id']` is assigning the ID of the newly created file object to the variable `train_id`.

When the `openai.File.create()` method is called, it returns a dictionary containing information about the newly created file object, including its ID. By accessing the `"id"` key of this dictionary using square bracket notation, we can retrieve the ID of the file object and assign it to the `train_id` variable.

This ID can then be used to reference the file object in subsequent API calls, such as when fine-tuning an NLP model using the uploaded training

```
train_id = train['id']
```

The code below uploads a file called "val.jsonl" and creates a file object using the OpenAI API. The purpose parameter is set to 'fine-tune', indicating that the file will be used for fine-tuning a language model.

To use this validation data for fine-tuning the GPT-3.5 model, you would typically pass the file object ID to the appropriate API method. However, it's important to note that there is no such thing as a GPT-3.5 model. The latest version of the GPT series is GPT-3, which has several different sizes (e.g. 13B, 6B, 2.7B, etc.) that correspond to the number of parameters in the model.

```
# Upload validation data
val = openai.File.create(
    file=open("val.jsonl", "rb"),
    purpose='fine-tune'
)
val
```

The code `val_id = val['id']` assigns the ID of the uploaded validation file to the variable `val_id`.

This is useful because the OpenAI API often requires file IDs as inputs for various methods, such as when fine-tuning a language model. By assigning the ID to a variable, you can easily reference it later in your code without having to remember the ID or look it up again.

```
val_id = val['id']
```

The code below creates a new fine-tuning job using the OpenAI API. Here's what each line does:

1. `response = openai.FineTuningJob.create(...)` - This line creates a new fine-tuning job using the `openai` Python module. The `create` method takes several parameters, including the ID of the training file (`train_id`), the ID of the validation file (`val_id`), the name of the model to use (`'gpt-3.5-turbo'`), and a suffix to add to the model name (`'an assistant'`).
2. `job_id = response["id"]` - This line extracts the ID of the newly created fine-tuning job from the `response` object and assigns it to the `job_id` variable.
3. `print("Job ID:", response["id"])` - This line prints the ID of the newly created fine-tuning job to the console.
4. `print("Status:", response["status"])` - This line prints the status of the newly created fine-tuning job to the console.

Overall, this code creates a new fine-tuning job for a GPT-3.5 model using a training file and a validation file, and prints the ID and status of the job to the console. The `suffix` parameter is used to give the model a custom name, which can be useful for keeping track of multiple fine-tuning jobs.

```

response = openai.FineTuningJob.create(
    training_file=train_id,
    validation_file=val_id,
    model="gpt-3.5-turbo",
    suffix="an assistant",
)
job_id = response["id"]
print("Job ID:", response["id"])
print("Status:", response["status"])
Job ID: ftjob-veBHcDc0U5W7EZr1iHFyXQu3
Status: validating_files

```

This line retrieves the details of a specific fine-tuning job using the `retrieve()` method provided by the OpenAI Python library. The `job_id` variable represents the unique identifier of the job you want to retrieve information about. The `response` variable will store the response received from the API.

```

response = openai.FineTuningJob.retrieve(job_id)

```

These lines print out specific details about the fine-tuning job.

The first line prints the Job ID, which is accessed from the `id` field in the response dictionary.

The second line prints the status of the job, which can be "in progress," "succeeded," or "failed." The status is obtained from the `status` field in the response dictionary.

The third line prints the number of trained tokens, which indicates the total number of tokens the fine-tuning job has processed so far. This information is retrieved from the `trained_tokens` field in the response dictionary.

```

print("Job ID:", response["id"])
print("Status:", response["status"])
print("Trained Tokens:", response["trained_tokens"])

```

```

Job ID: ftjob-veBHcDc0U5W7EZr1iHFyXQu3
Status: succeeded
Trained Tokens: 29316

```

The code below retrieves the events associated with a specific fine-tuning job using the OpenAI API. Here's what each line does:

1. ``response = openai.FineTuningJob.list_events(id=job_id, limit=50)`` - This line retrieves the events associated with the fine-tuning job with the ID ``job_id``. The ``list_events`` method takes several parameters, including the ID of the job to retrieve events for (``id``) and the maximum number of events to retrieve (``limit``).

2. ``events = response["data"]`` - This line extracts the list of events from the ``response`` object and assigns it to the ``events`` variable.

3. ``events.reverse()`` - This line reverses the order of the events so that the most recent event is printed first.

4. ``for event in events:`` - This line starts a loop that iterates over each event in the ``events`` list.

5. ``print(event["message"])`` - This line prints the message associated with each event to the console.

Overall, this code retrieves the events associated with a specific fine-tuning job and prints the message associated with each event to the console. This can be useful for monitoring the progress of a fine-tuning job and identifying any issues that may have occurred.

```

response = openai.FineTuningJob.list_events(id=job_id, limit=50)

events = response["data"]
events.reverse()

for event in events:
    print(event["message"])

```

Created fine-tuning job: ftjob-veBHcDc0U5W7EZr1iHFyXQu3
 Validating training file: file-BZEIxxYsUwHPX81ec6HIFdRY
 Files validated, moving job to queued state
 Fine-tuning job started
 Step 1/240: training loss=1.34
 Step 11/240: training loss=0.56
 Step 21/240: training loss=0.50
 Step 31/240: training loss=0.31
 Step 41/240: training loss=0.30
 Step 51/240: training loss=0.06
 Step 61/240: training loss=0.37
 Step 71/240: training loss=0.13
 Step 81/240: training loss=0.10
 Step 91/240: training loss=0.26
 Step 101/240: training loss=0.29
 Step 111/240: training loss=0.34
 Step 121/240: training loss=0.09
 Step 131/240: training loss=0.09
 Step 141/240: training loss=0.07
 Step 151/240: training loss=0.23
 Step 161/240: training loss=0.06
 Step 171/240: training loss=0.02
 Step 181/240: training loss=0.07
 Step 191/240: training loss=0.15
 Step 201/240: training loss=0.12
 Step 211/240: training loss=0.09
 Step 221/240: training loss=0.01
 Step 231/240: training loss=0.12
 New fine-tuned model created: ft:gpt-3.5-turbo-0613:chattech-corp:an-assistant:82KU5Mx0
 The job has successfully completed

The code below retrieves the ID of the fine-tuned model associated with a specific fine-tuning job using the OpenAI API. Here's what each line does:

1. ``response = openai.FineTuningJob.retrieve(job_id)`` - This line retrieves information about the fine-tuning job with the ID ``job_id`` using the ``retrieve`` method of the ``FineTuningJob`` class.
2. ``fine_tuned_model_id = response["fine_tuned_model"]`` - This line extracts the ID of the fine-tuned model from the ``response`` object and assigns it to the ``fine_tuned_model_id`` variable.
3. ``if fine_tuned_model_id is None:`` - This line checks if the ``fine_tuned_model_id`` variable is ``None``, which would indicate that the fine-tuning job has not yet completed.
4. ``raise RuntimeError("Fine-tuned model ID not found. Your job has likely not been completed yet.")`` - This line raises a ``RuntimeError`` with an error message if the ``fine_tuned_model_id`` variable is ``None``.
5. ``print("Fine-tuned model ID:", fine_tuned_model_id)`` - This line prints the ID of the fine-tuned model to the console.

Overall, this code retrieves the ID of the fine-tuned model associated with a specific fine-tuning job and checks if the fine-tuning job has completed. If the fine-tuning job has not completed, an error is raised. If the fine-tuning job has completed, the ID of the fine-tuned model is printed to the console. This can be useful for retrieving the ID of a fine-tuned model for further use in other applications.

```

response = openai.FineTuningJob.retrieve(job_id)
fine_tuned_model_id = response["fine_tuned_model"]

if fine_tuned_model_id is None:
    raise RuntimeError("Fine-tuned model ID not found. Your job has likely not been
completed yet.")

print("Fine-tuned model ID:", fine_tuned_model_id)
Fine-tuned model ID: ft:gpt-3.5-turbo-0613:chattech-corp:an-assistant:82KU5Mx0

```

1. Real-time Data Integration:

The last step is to use your fine-tuned model for inference. you simply call `ChatCompletions` with your new fine-tuned model name filling the `model` parameter.

The code defines a list of test messages for your AI chatbot. The test messages consist of a system message and a user message.

The system message provides some information about the chatbot's role and how it should respond to user requests. It also sets the tone for the conversation by asking the chatbot to use polite and respectful language.

The user message is an example of a request that the chatbot might receive. It asks the chatbot to find hotels in Cairo with availability for 10 guests, a 5-star rating, and a preferred budget of \$1000 on September 29th, 2023.

Defining test messages like this can be useful for testing and evaluating the performance of your chatbot. You can use these messages to simulate conversations with your chatbot and see how well it responds to different types of requests and inputs.

```

test_messages = [
    {"role": "system", "content": "You are AI Assistant that tailors vacation
plans. Please reply user requests using polite and respectful language."},
    {"role": "user", "content": "Find hotels in cairo with availability, my
preferred budget is 1000$, the number of guests are 10, 5 star rating, on
29/9/2023."}
]

```

The code below uses the OpenAI API to generate a chat response using a fine-tuned language model. Here's what each line does:

1. `response = openai.ChatCompletion.create(model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500)` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the fine-tuned model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation. The `temperature` parameter controls the randomness of the generated response, with a value of 0 indicating that the response should be deterministic. The `max_tokens` parameter specifies the maximum number of tokens (words or subwords) in the generated response.

2. `print(response["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `response` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using a fine-tuned language model and prints the response to the console. This can be useful for building chatbots or other conversational AI applications. Note that the quality of the generated response will depend on the quality of the fine-tuned model and the input messages provided.

```
response = openai.ChatCompletion.create(
    model=fine_tuned_model_id, messages=test_messages, temperature=0,
    max_tokens=500
)
print(response["choices"][0]["message"]["content"])
```

Response of AI Assistant

I'm sorry, but I am unable to browse the internet or check real-time availability. However, you can use online travel agencies or hotel booking websites to find hotels in Cairo that meet your criteria.

The code below uses the OpenAI API to generate a chat response using the GPT-3.5 Turbo language model. Here's what each line does:

1. `completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=test_messages)` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the GPT-3.5 Turbo language model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation.
2. `print(completion["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `completion` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using the GPT-3.5 Turbo language model and prints the response to the console. The quality of the generated response will depend on the quality of the input messages provided and the capabilities of the GPT-3.5 Turbo model.

```
completion = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=test_messages
)
print(completion["choices"][0]["message"]["content"])
```

Response of AI Assistant

Certainly! I'd be happy to help you with that. I have found a few hotels in Cairo that match your criteria. Here are a couple of options:

1. Hotel 1: XYZ Hotel
 - Rating: 5 stars
 - Price: \$1000 per night
 - Availability: Available on 29/9/2023
 - Room capacity: Can accommodate up to 10 guests
2. Hotel 2: ABC Hotel
 - Rating: 5 stars
 - Price: \$1000 per night
 - Availability: Available on 29/9/2023
 - Room capacity: Can accommodate up to 10 guests

Please note that the prices provided are an estimate and may vary slightly depending on the specific dates and any additional services you may require. I recommend contacting the hotels directly to confirm availability and make your reservation.

2. Complex Query Handling:

The code defines a list of test messages for your AI chatbot. The test messages consist of a system message and a user message.

The system message provides some information about the chatbot's role and how it should respond to user requests. It also sets the tone for the conversation by asking the chatbot to use polite and respectful language.

The user message is an example of a request that the chatbot might receive. It asks the chatbot the following: I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under \$1000, in Egypt.

Defining test messages like this can be useful for testing and evaluating the performance of your chatbot. You can use these messages to simulate conversations with your chatbot and see how well it responds to different types of requests and inputs.

```
test_messages = [
    {"role": "system", "content": "You are AI Assistant that tailors vacation plans. Please reply user requests using polite and respectful language."},
    {"role": "user", "content": "I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under $1000, in Egypt"}
]
```

The code below uses the OpenAI API to generate a chat response using a fine-tuned language model. Here's what each line does:

1. `response = openai.ChatCompletion.create(model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500)` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the fine-tuned model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation. The `temperature` parameter controls the randomness of the generated response, with a value of 0 indicating that the response should be deterministic. The `max_tokens` parameter specifies the maximum number of tokens (words or subwords) in the generated response.

2. `print(response["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `response` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using a fine-tuned language model and prints the response to the console. This can be useful for building chatbots or other conversational AI applications. Note that the quality of the generated response will depend on the quality of the fine-tuned model and the input messages provided.

```
completion1= openai.ChatCompletion.create(
    model=fine_tuned_model_id, messages=test_messages, temperature=0,
    max_tokens=500
)
print(completion1["choices"][0]["message"]["content"])
```

I recommend spending 2 days in Hurghada for beach relaxation and snorkeling, then visiting Luxor for its historical sites like Karnak Temple and the Valley of the Kings. This itinerary should fit your budget, including flights and accommodation.

The code below uses the OpenAI API to generate a chat response using the GPT-3.5 Turbo language model. Here's what each line does:

1. `completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=test_messages)`` - This line creates a chat completion request using the `create`` method of the `ChatCompletion`` class. The `model`` parameter specifies the ID of the GPT-3.5 Turbo language model to use for the chat response. The `messages`` parameter is a list of previous messages in the conversation.
2. `print(completion["choices"][0]["message"]["content"])`` - This line prints the generated chat response to the console. The response is obtained from the `completion`` object using the `choices`` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]``. The content of the message is obtained from the `message`` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using the GPT-3.5 Turbo language model and prints the response to the console. The quality of the generated response will depend on the quality of the input messages provided and the capabilities of the GPT-3.5 Turbo model.

```
completion2 = openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages=test_messages  
)  
print(completion2["choices"][0]["message"]["content"])
```

Certainly! I can help you plan a 4-day trip in Egypt that combines beach relaxation, historical sites, and a touch of adventure within your budget. Here's an itinerary suggestion:

Day 1: Arrive in Cairo

- Upon arrival at Cairo International Airport, transfer to your hotel.
- Take some time to relax and acclimate to the new surroundings.

Day 2: Explore Cairo

- Start your day with a visit to the iconic Giza Pyramids, including the Great Pyramid of Giza, Sphinx, and the Valley Temple.
- In the afternoon, explore the Egyptian Museum to marvel at artifacts from ancient Egypt, including the treasures of Tutankhamun.
- Wrap up your day with a visit to Khan El Khalili Bazaar for a taste of traditional shopping and local experiences.

Day 3: Hurghada Beach

- Transfer to Hurghada, a beautiful coastal city on the Red Sea.
- Relax and enjoy the pristine beaches, indulge in water sports activities like snorkeling or scuba diving, or simply unwind under the sun.
- If you're feeling adventurous, you can also opt for a desert safari or quad biking experience.

Day 4: Luxor Day Trip

- Take a day trip from Hurghada to Luxor, known as the "world's greatest open-air museum."
- Explore the Valley of the Kings, where ancient Egyptian pharaohs were buried, including Tutankhamun's tomb.
- Visit the stunning Karnak Temple Complex and Luxor Temple, both filled with magnificent ancient Egyptian structures and hieroglyphics.
- After the tour, transfer back to Hurghada.

Please keep in mind that the prices for flights and accommodation may vary based on the time of year and the current promotions. It's always recommended to book in advance to secure the best deals within your budget. Let me know if there's anything else I can assist you with!

3. Safety & Local Guidelines:

The code defines a list of test messages for your AI chatbot. The test messages consist of a system message and a user message.

The system message provides some information about the chatbot's role and how it should respond to user requests. It also sets the tone for the conversation by asking the chatbot to use polite and respectful language.

The user message is an example of a request that the chatbot might receive. It asks the chatbot to Provide safety tips for luxor.

Defining test messages like this can be useful for testing and evaluating the performance of your chatbot. You can use these messages to simulate conversations with your chatbot and see how well it responds to different types of requests and inputs.

```
test_messages = [
    {"role": "system", "content": "You are AI Assistant that tailors vacation
plans. Please reply user requests using polite and respectful language."},
    {"role": "user", "content": "Provide safety tips for luxor."}
]
```

The code below uses the OpenAI API to generate a chat response using a fine-tuned language model. Here's what each line does:

1. `response = openai.ChatCompletion.create(model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500)` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the fine-tuned model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation. The `temperature` parameter controls the randomness of the generated response, with a value of 0 indicating that the response should be deterministic. The `max_tokens` parameter specifies the maximum number of tokens (words or subwords) in the generated response.

2. `print(response["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `response` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using a fine-tuned language model and prints the response to the console. This can be useful for building chatbots or other conversational AI applications. Note that the quality of the generated response will depend on the quality of the fine-tuned model and the input messages provided.

```
completion5= openai.ChatCompletion.create(
    model=fine_tuned_model_id, messages=test_messages, temperature=0,
max_tokens=500
)
print(completion5["choices"][0]["message"]["content"])
```

While Luxor is generally safe for tourists, it's advisable to take precautions like avoiding isolated areas at night, using licensed taxis, and being aware of your surroundings. It's also recommended to follow any guidance or instructions from local authorities or your tour guide.

The code below uses the OpenAI API to generate a chat response using the GPT-3.5 Turbo language model. Here's what each line does:

1. `completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=test_messages)`` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the GPT-3.5 Turbo language model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation.
2. `print(completion["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `completion` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using the GPT-3.5 Turbo language model and prints the response to the console. The quality of the generated response will depend on the quality of the input messages provided and the capabilities of the GPT-3.5 Turbo model.

```
completion6 = openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages=test_messages  
)  
print(completion6["choices"][0]["message"]["content"])
```

Sure! Here are some safety tips for your visit to Luxor:

1. Research and plan: Before your trip, take the time to research the local customs, laws, and any travel advisories issued for Luxor. This will help you stay informed about any potential safety concerns.
2. Dress appropriately: Luxor is a conservative city, and it is respectful to dress modestly, especially when visiting religious sites. Make sure to cover your shoulders and knees, and avoid wearing revealing or offensive clothing.
3. Avoid public displays of wealth: To minimize the risk of theft or attracting unwanted attention, it's advisable to avoid displaying expensive items like jewelry, cameras, or large amounts of cash in public.
4. Use reliable transportation: Opt for licensed taxis or arrange transportation through your hotel or a reputable tour operator. Avoid getting into unmarked or unofficial taxis to ensure your safety.
5. Be cautious with street vendors: While exploring the local markets, be mindful of your personal belongings and valuables. Some vendors may employ aggressive selling techniques, so it's important to remain assertive but polite. Also, beware of counterfeit goods and negotiate prices reasonably.
6. Stay hydrated and protect yourself from the sun: Luxor experiences high temperatures, especially during summer months. It's crucial to stay hydrated, drink bottled water, and use sunscreen and protective clothing to avoid heatstroke or sunburn.
7. Follow local customs and laws: Respect the local customs and traditions, including religious practices. Remember to ask for permission before taking photos of people and be aware of any restricted or prohibited areas.
8. Stay in well-lit and populated areas: While Luxor is generally safe, it's advisable to stick to well-populated areas and avoid walking alone late at night. If you're unsure about a particular location, consult with your hotel staff or local authorities.

Remember, these tips will help enhance your safety, but it's essential to trust your instincts and be aware of your surroundings at all times. Enjoy your trip to Luxor!

Another Example for Safety & Local Guidelines:

The code defines a list of test messages for your AI chatbot. The test messages consist of a system message and a user message.

The system message provides some information about the chatbot's role and how it should respond to user requests. It also sets the tone for the conversation by asking the chatbot to use polite and respectful language.

The user message is an example of a request that the chatbot might receive. It asks the chatbot `{"role": "user", "content": "What are the local customs in new cairo?"}`

Defining test messages like this can be useful for testing and evaluating the performance of your chatbot. You can use these messages to simulate conversations with your chatbot and see how well it responds to different types of requests and inputs.

```
test_messages = [
    {"role": "system", "content": "You are AI Assistant that tailors vacation plans. Please reply user requests using polite and respectful language."},
    {"role": "user", "content": "What are the local customs in new cairo?"}
]
```

The code below uses the OpenAI API to generate a chat response using a fine-tuned language model. Here's what each line does:

1. ``response = openai.ChatCompletion.create(model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500)`` - This line creates a chat completion request using the ``create`` method of the ``ChatCompletion`` class. The ``model`` parameter specifies the ID of the fine-tuned model to use for the chat response. The ``messages`` parameter is a list of previous messages in the conversation. The ``temperature`` parameter controls the randomness of the generated response, with a value of 0 indicating that the response should be deterministic. The ``max_tokens`` parameter specifies the maximum number of tokens (words or subwords) in the generated response.

2. ``print(response["choices"][0]["message"]["content"])`` - This line prints the generated chat response to the console. The response is obtained from the ``response`` object using the ``choices`` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using ``[0]``. The content of the message is obtained from the ``message`` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using a fine-tuned language model and prints the response to the console. This can be useful for building chatbots or other conversational AI applications. Note that the quality of the generated response will depend on the quality of the fine-tuned model and the input messages provided

```
completion7= openai.ChatCompletion.create(
    model=fine_tuned_model_id, messages=test_messages, temperature=0,
max_tokens=500
)
print(completion7["choices"][0]["message"]["content"])
```

New Cairo is a modern city in Egypt with diverse cultures. However, it is generally respectful to dress modestly and adhere to Islamic customs.

The code below uses the OpenAI API to generate a chat response using the GPT-3.5 Turbo language model. Here's what each line does:

1. `completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=test_messages)`` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the GPT-3.5 Turbo language model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation.
2. `print(completion["choices"][0]["message"]["content"])`` - This line prints the generated chat response to the console. The response is obtained from the `completion` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using the GPT-3.5 Turbo language model and prints the response to the console. The quality of the generated response will depend on the quality of the input messages provided and the capabilities of the GPT-3.5 Turbo model.

```
completion8 = openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages=test_messages  
)  
  
print(completion8["choices"][0]["message"]["content"])
```

In New Cairo, you can expect to encounter a unique blend of customs and traditions. Here are a few local customs that you may want to be aware of:

1. **Respect for Religion:** Egypt is an Islamic country, and therefore, it is important to respect Islamic customs and traditions. Modest attire is appreciated, especially when visiting religious sites.
2. **Greeting Etiquette:** When meeting someone for the first time, a handshake is customary. It is also polite to say "As-salamu alaykum" (peace be upon you) as a greeting.
3. **Punctuality:** Egyptians generally have a more relaxed approach to timekeeping, so it is best to be flexible and not expect strict punctuality. However, it is recommended to be on time for formal occasions and business meetings.
4. **Social Etiquette:** Egyptians are known for their warm and hospitable nature. It is common for guests to be greeted with tea or coffee, and it is polite to accept such offers.
5. **Nonverbal Communication:** Nonverbal gestures, such as pointing with fingers, can be considered impolite in Egyptian culture. Instead, it is preferred to use an open hand in a downward direction to indicate something.

Remember, immersing yourself in local customs can enhance your travel experience and show respect for the local culture. Enjoy your time in New Cairo!

4. Multi-language Support:

The code defines a list of test messages for your AI chatbot. The test messages consist of a system message and a user message.

The system message provides some information about the chatbot's role and how it should respond to user requests. It also sets the tone for the conversation by asking the chatbot to use polite and respectful language.

The user message is an example of a request that the chatbot might receive. It asks the chatbot to Provide safety tips for luxor.

Defining test messages like this can be useful for testing and evaluating the performance of your chatbot. You can use these messages to simulate conversations with your chatbot and see how well it responds to different types of requests and inputs.

```
test_messages = [
    {"role": "system", "content": "You are AI Assistant that tailors vacation plans. Please reply user requests using polite and respectful language."},
    {"role": "user", "content": "Ich möchte eine 10-tägige Reise mit einer Mischung aus Strandentspannung und historischen Stätten für weniger als 2.000 US-Dollar."}
]
```

The code below uses the OpenAI API to generate a chat response using a fine-tuned language model. Here's what each line does:

1. `response = openai.ChatCompletion.create(model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500)` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the fine-tuned model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation. The `temperature` parameter controls the randomness of the generated response, with a value of 0 indicating that the response should be deterministic. The `max_tokens` parameter specifies the maximum number of tokens (words or subwords) in the generated response.

2. `print(response["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `response` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using a fine-tuned language model and prints the response to the console. This can be useful for building chatbots or other conversational AI applications. Note that the quality of the generated response will depend on the quality of the fine-tuned model and the input messages provided

```
completion9= openai.ChatCompletion.create(
    model=fine_tuned_model_id, messages=test_messages, temperature=0,
max_tokens=500
)
print(completion9["choices"][0]["message"]["content"])
```

Wie wäre es mit einer Reise nach Griechenland? Sie können einige Tage auf den schönen Inseln entspannen und dann historische Stätten wie Athen und Delphi besuchen.

The code below uses the OpenAI API to generate a chat response using the GPT-3.5 Turbo language model. Here's what each line does:

1. `completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=test_messages)` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the GPT-3.5 Turbo language model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation.
2. `print(completion["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `completion` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using the GPT-3.5 Turbo language model and prints the response to the console. The quality of the generated response will depend on the quality of the input messages provided and the capabilities of the GPT-3.5 Turbo model.

```
completion10= openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages=test_messages  
)  
  
print(completion10["choices"][0]["message"]["content"])
```

Natürlich! Ich kann Ihnen bei der Planung Ihrer 10-tägigen Reise mit einer Mischung aus Strandentspannung und historischen Stätten helfen. Basierend auf Ihrem Budget von weniger als 2.000 US-Dollar können wir verschiedene Optionen erkunden. Welches Reiseziel interessiert Sie?

5. Continuous Feedback Loop:

The code defines a list of test messages for your AI chatbot. The test messages consist of a system message and a user message.

The system message provides some information about the chatbot's role and how it should respond to user requests. It also sets the tone for the conversation by asking the chatbot to use polite and respectful language.

The user message is an example of a request that the chatbot might receive. It asks the chatbot Were the local activities enjoyable in new cairo?

Defining test messages like this can be useful for testing and evaluating the performance of your chatbot. You can use these messages to simulate conversations with your chatbot and see how well it responds to different types of requests and inputs.

```
test_messages = [  
    {"role": "system", "content": "You are AI Assistant that tailors vacation  
plans. Please reply user requests using polite and respectful language."},  
    {"role": "user", "content": "Were the local activities enjoyable in new  
cairo?"}  
]
```

The code below uses the OpenAI API to generate a chat response using a fine-tuned language model. Here's what each line does:

1. ``response = openai.ChatCompletion.create(model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500)`` - This line creates a chat completion request using the ``create`` method of the ``ChatCompletion`` class. The ``model`` parameter specifies the ID of the fine-tuned model to use for the chat response. The ``messages`` parameter is a list of previous messages in the conversation. The ``temperature`` parameter controls the randomness of the generated response, with a value of 0 indicating that the response should be deterministic. The ``max_tokens`` parameter specifies the maximum number of tokens (words or subwords) in the generated response.

2. ``print(response["choices"][0]["message"]["content"])`` - This line prints the generated chat response to the console. The response is obtained from the ``response`` object using the ``choices`` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using ``[0]``. The content of the message is obtained from the ``message`` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using a fine-tuned language model and prints the response to the console. This can be useful for building chatbots or other conversational AI applications. Note that the quality of the generated response will depend on the quality of the fine-tuned model and the input messages provided

```
completion11= openai.ChatCompletion.create(  
    model=fine_tuned_model_id, messages=test_messages, temperature=0,  
max_tokens=500  
)  
print(completion11["choices"][0]["message"]["content"])
```

New Cairo offers a variety of enjoyable activities such as visiting parks, shopping malls, and trying out different restaurants and cafes.

The code below uses the OpenAI API to generate a chat response using the GPT-3.5 Turbo language model. Here's what each line does:

1. ``completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=test_messages)`` - This line creates a chat completion request using the ``create`` method of the ``ChatCompletion`` class. The ``model`` parameter specifies the ID of the GPT-3.5 Turbo language model to use for the chat response. The ``messages`` parameter is a list of previous messages in the conversation.

2. ``print(completion["choices"][0]["message"]["content"])`` - This line prints the generated chat response to the console. The response is obtained from the ``completion`` object using the ``choices`` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using ``[0]``. The content of the message is obtained from the ``message`` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using the GPT-3.5 Turbo language model and prints the response to the console. The quality of the generated response will depend on the quality of the input messages provided and the capabilities of the GPT-3.5 Turbo model.

```
completion12= openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages=test_messages  
)  
print(completion12["choices"][0]["message"]["content"])
```

Yes, most visitors find the local activities in New Cairo quite enjoyable. The city offers a wide range of activities to suit different interests and preferences. From exploring local markets and shopping centers to dining at a variety of restaurants, playing sports at recreational clubs, visiting parks and gardens, or even indulging in spa and wellness treatments, there is something for everyone. Additionally, New Cairo is known for its vibrant nightlife, featuring bars, lounges, and entertainment venues. Whether you are looking for cultural experiences or simply want to unwind and have fun, New Cairo has much to offer.

6. Budgetary Optimization:

The code defines a list of test messages for your AI chatbot. The test messages consist of a system message and a user message.

The system message provides some information about the chatbot's role and how it should respond to user requests. It also sets the tone for the conversation by asking the chatbot to use polite and respectful language.

The user message is an example of a request that the chatbot might receive. It asks the chatbot to suggest best-value options for accommodations, activities, and dining, with budget 300\$, in port said.

Defining test messages like this can be useful for testing and evaluating the performance of your chatbot. You can use these messages to simulate conversations with your chatbot and see how well it responds to different types of requests and inputs.

```
test_messages = [
    {"role": "system", "content": "You are AI Assistant that tailors vacation plans. Please reply user requests using polite and respectful language."},
    {"role": "user", "content": "suggest best-value options for accommodations, activities, and dining, with budget 300$, in port said"}
]
```

The code below uses the OpenAI API to generate a chat response using a fine-tuned language model. Here's what each line does:

1. ``response = openai.ChatCompletion.create(model=fine_tuned_model_id, messages=test_messages, temperature=0, max_tokens=500)`` - This line creates a chat completion request using the ``create`` method of the ``ChatCompletion`` class. The ``model`` parameter specifies the ID of the fine-tuned model to use for the chat response. The ``messages`` parameter is a list of previous messages in the conversation. The ``temperature`` parameter controls the randomness of the generated response, with a value of 0 indicating that the response should be deterministic. The ``max_tokens`` parameter specifies the maximum number of tokens (words or subwords) in the generated response.

2. ``print(response["choices"][0]["message"]["content"])`` - This line prints the generated chat response to the console. The response is obtained from the ``response`` object using the ``choices`` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using ``[0]``. The content of the message is obtained from the ``message`` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using a fine-tuned language model and prints the response to the console. This can be useful for building chatbots or other conversational AI applications. Note that the quality of the generated response will depend on the quality of the fine-tuned model and the input messages provided.


```
completion13= openai.ChatCompletion.create(  
    model=fine_tuned_model_id, messages=test_messages, temperature=0,  
max_tokens=500  
)  
print(completion13["choices"][0]["message"]["content"])
```

For accommodations, you can consider budget-friendly hotels like Port Said Hotel or Resta Port Said Hotel. As for activities, you can explore the Port Said Lighthouse and visit the Port Said National Museum. For dining, local eateries like El Manara and El Kababgy offer affordable and delicious meals.

The code below uses the OpenAI API to generate a chat response using the GPT-3.5 Turbo language model. Here's what each line does:

1. `completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=test_messages)` - This line creates a chat completion request using the `create` method of the `ChatCompletion` class. The `model` parameter specifies the ID of the GPT-3.5 Turbo language model to use for the chat response. The `messages` parameter is a list of previous messages in the conversation.
2. `print(completion["choices"][0]["message"]["content"])` - This line prints the generated chat response to the console. The response is obtained from the `completion` object using the `choices` key, which contains a list of possible completions. In this case, we are using the first (and only) completion in the list, which is accessed using `[0]`. The content of the message is obtained from the `message` key, which contains a dictionary with information about the generated message, including the text content.

Overall, this code generates a chat response using the GPT-3.5 Turbo language model and prints the response to the console. The quality of the generated response will depend on the quality of the input messages provided and the capabilities of the GPT-3.5 Turbo model.

```
completion14= openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages=test_messages  
)  
print(completion14["choices"][0]["message"]["content"])
```

Thank you for reaching out! Port Said offers a variety of options to suit your budget of \$300. Here are some suggestions for accommodations, activities, and dining:

Accommodations:

1. Local guesthouses: Consider staying in local guesthouses or budget-friendly hotels such as Statica Hotel or El Nouzha Hotel.
2. Hostels: Look into Port Said Youth Hostel or other budget-friendly hostels which offer affordable dormitory-style accommodations.
3. Vacation rentals: Explore platforms like Airbnb or Booking.com to find cozy apartments or studios that fit your budget.

Activities:

1. Visit Port Said National Museum: Immerse yourself in the history and culture of the city at this museum, which showcases a diverse range of artifacts.
2. Explore the Suez Canal: Take a boat tour along the Suez Canal for an up-close view of the impressive ships and containers passing through.
3. Wander along the Corniche: Enjoy a leisurely stroll along Port Said's Corniche, offering beautiful views of the Mediterranean Sea and the city's iconic lighthouse.

Dining:

1. Local street food: Experience the vibrant culinary scene of Port Said by trying affordable street food options such as falafel, koshary, or seafood sandwiches from local vendors.
2. Affordable restaurants: Check out restaurants like Al Dabke or El Mehshi, which offer delicious Egyptian cuisine at reasonable prices.
3. Cafes and bakeries: Enjoy snacks and pastries from local cafes and bakeries like Aboulfaouz or Al Sultan, where you can savor Egyptian sweets and Middle Eastern treats.

Please note that prices may vary, so it's always a good idea to check and compare options in advance. Have a wonderful vacation in Port Said!

7. Dynamic Profile Creation:

This is a Python script that uses the OpenAI API and the Hugging Face Datasets library to create a travel itinerary generator. Here's a breakdown of what the code does:

1. The script imports the `openai` and `datasets` modules and sets the OpenAI API key to an empty string. You will need to replace the empty string with your actual OpenAI API key to use the script.
2. The script loads a travel dataset from the Hugging Face Datasets library. The dataset contains information about travel destinations, including tags (e.g., beach, mountains, culture, adventure), price, and food options.
3. The `create_profile` function prompts the user for their personal details, including name, age, gender, email, current city, travel history, interests, dietary restrictions, and budget. The function returns a dictionary containing the user's details.
4. The `generate_itinerary` function takes the user's details as input and uses the OpenAI API to generate an optimized travel itinerary based on the user's preferences. The function constructs a prompt string using the user's details and calls the OpenAI API to generate a response. The function returns the generated itinerary as a string.
5. The `main` function calls `create_profile` to get the user's details, calls `generate_itinerary` to generate an optimized itinerary based on the user's preferences, and prints the generated itinerary to the console.
6. The script checks if the `__name__` variable is equal to ` "__main__"`, which indicates that the script is being run as the main program. If so, the script calls the `main` function.

Overall, this script demonstrates how to use the OpenAI API and the Hugging Face Datasets library to create a travel itinerary generator that can generate personalized travel itineraries based on the user's preferences.

```
import openai
from datasets import load_dataset

openai.api_key = "sk-53ANYjjA7L8oaXvTbWBRT3BIbkFJKQslm6R0nNDTulrZtrSt"

ds = load_dataset("thanhnew2001/travel",split="train")

def create_profile():
    name = input("What is your name?")
    age = input("What is your age?")
    gender = input("What is your gender?")
    email = input("What is your email address?")

    current_city =input("what is your current city?")

    travel_history = input("Where have you traveled before?")

    interests = input("What are your interests? (beach, mountains, culture, adventure, etc.)")

    dietary_restrictions = input("Do you have any dietary restrictions?")
```

```

budget = input("What is your budget for this trip?")

return {
    "current_city":current_city,
    "travel_history": travel_history,
    "interests": interests,
    "dietary_restrictions": dietary_restrictions,
    "budget": budget
}

def generate_itinerary(user_details):
    prompt = f"Generate an optimized itinerary for a {user_details['travel_history']} travel_history that includes,
{user_details['current_city']} current_city that includes, {user_details['budget']} budget that includes
{user_details['interests']} activities and accommodates {user_details['dietary_restrictions']} dietary restrictions."
    response = openai.Completion.create(
        engine="text-davinci-002",
        prompt=prompt,
        max_tokens=1024,
        n=1,
        stop=None,
        temperature=0.5,
    )
    itinerary = response.choices[0].text

    return itinerary

filtered_dataset = ds.filter(lambda example: interests in example["tags"] and example["price"] <= budget and
dietary_restrictions not in example["food"])

print(filtered_dataset)

def main():
    user_details = create_profile()

    itinerary = generate_itinerary(user_details)

    print("Here's your optimized itinerary:")
    print(itinerary)

if __name__ == "__main__":
    main()

```

What is your name?ahmed gomaa
What is your age?27
What is your gender?male
What is your email address? ahmedgomaa44@gmail.com
what is your current city?cairo
Where have you traveled before?paris
What are your interests? (beach, mountains, culture, adventure, etc.)culture
Do you have any dietary restrictions?no
What is your budget for this trip?3000 egp
Here's your optimized itinerary:

Day 1:

Arrive in Paris and check into your hotel.

Spend the morning exploring the city on foot. Visit the Eiffel Tower, the Louvre Museum, and Notre Dame Cathedral.

In the afternoon, take a boat ride on the Seine River.

In the evening, enjoy a romantic dinner at a charming Parisian café.

Day 2:

Spend the day touring the Palace of Versailles.

In the afternoon, take a leisurely stroll through the gardens of the Tuileries.

In the evening, enjoy a delicious French dinner followed by a show at the Moulin Rouge.

Day 3:

Depart for Cairo and check into your hotel.

Spend the day exploring the city, including the Egyptian Museum and the Great Pyramids of Giza.

In the evening, enjoy a traditional Egyptian dinner followed by a belly dancing show.

Day 4:

Spend the day relaxing by the pool or exploring the city on your own.

In the evening, enjoy a delicious dinner at a local restaurant.

Day 5:

Depart for home.

8. Interactive Itinerary Adjustments:

This is a Python script that uses the OpenAI API to create a travel assistant chatbot. Here's a breakdown of what the code does:

1. The script imports the `openai` module and sets the OpenAI API key to an empty string. You will need to replace the empty string with your actual OpenAI API key to use the script.
2. The `generate_response` function takes a prompt string as input, calls the OpenAI API to generate a response based on the prompt, and returns the generated response as a string.
3. The `handle_input` function takes user input as input, checks if the input contains certain keywords (e.g., "trip", "change", "thanks", "bye"), and generates a response based on the input using the `generate_response` function. If the input does not contain any of the keywords, the function generates a default response.
4. The `start_conversation` function prints a welcome message and prompts the user for input. The function then calls the `handle_input` function to generate a response based on the user's input and prints the response to the console. The function continues to prompt the user for input and generate responses until the user inputs "goodbye" or the chatbot generates a response containing "have a great trip".
5. The script calls the `start_conversation` function to start the chatbot.

```
import openai

openai.api_key = "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"

def generate_response(prompt):
    response = openai.Completion.create(
        engine="text-davinci-002",
        prompt=prompt,
        max_tokens=1024,
        n=1,
        stop=None,
        temperature=0.5,
    )
    return response.choices[0].text.strip()

def handle_input(user_input):
    if "trip" in user_input.lower():
        prompt = f"I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under $3000, in Paris"
        response = generate_response(prompt)
        return response
    elif "change" in user_input.lower():
        prompt = f"Suggest a local festival for trip to paris"
        response = generate_response(prompt)
        return response
    elif "thanks" in user_input.lower() or "thank you" in user_input.lower():
        prompt = f"Thank you"
        response = generate_response(prompt)
```

```

        return response

    elif "bye" in user_input.lower() or "goodbye" in user_input.lower():

        prompt = f"goodbye!"

        response = generate_response(prompt)

        return response

    else:

        prompt = f"I'm sorry, I didn't understand. Can you please rephrase your request?"

        response = generate_response(prompt)

        return response

def start_conversation():

    print("Welcome to our travel assistant! How can I help you today?")

    while True:

        user_input = input("> ")

        response = handle_input(user_input)

        print(response)

        if "goodbye" in user_input.lower() or "have a great trip" in response.lower():

            break

start_conversation()

```

Welcome to our travel assistant! How can I help you today?

> I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under \$3000, in Paris

This is a great trip for anyone who wants to experience a little bit of everything that Paris has to offer. Day 1 can be spent exploring the historical sites of the city, such as the Eiffel Tower, the Louvre, and Notre Dame Cathedral. Day 2 can be spent relaxing at one of the many beaches in the city, such as the Plage de la Concorde or the Plage de la Bastille. Day 3 can be spent exploring the many shops and restaurants in the city, and Day 4 can be spent on an adventure, such as hiking in the nearby forests or visiting one of the many castles in the area.

> Suggest a local festival for trip to paris

Day 1: Arrive in Paris and check into your hotel. Spend the day exploring the city on your own, perhaps visiting the Eiffel Tower, the Louvre Museum, or Notre Dame Cathedral.

Day 2: Take a day trip to the Normandy beaches, where you can explore the history of D-Day and the Battle of Normandy.

Day 3: Head to the Palace of Versailles to see the former home of French kings and queens.

Day 4: Spend your final day in Paris enjoying some of the city's many parks and gardens, such as the Jardin des Tuileries or the Luxembourg Gardens.

> Thank you

for your time.

Sincerely,

[Your Name]

[Your Address]

[City, State ZIP]

[Your Email]

> goodbye!

The first time I ever heard the phrase “goodbye, my love” was in a song by the Beatles. The song is called “Yesterday” and it is a beautiful ballad about a man who is reminiscing about a lost love. The phrase “goodbye, my love” is used twice in the song, once when the man is saying goodbye to his love and once when he is talking about how much he misses her. It is a sad song, but it is also a very beautiful song.

Another Solution for Interactive Itinerary Adjustments:

This is a Python script that uses the OpenAI API to create a chatbot that can assist with travel planning. Here's a breakdown of what the code does:

1. The script imports the `openai` module and sets the API key to an empty string. You will need to replace the empty string with your actual OpenAI API key to use the script.
2. The `generate_response` function takes a `prompt` string as input and uses the OpenAI API to generate a response to the prompt. The function specifies the `engine` to use (`text-davinci-002`), the maximum number of tokens in the response (`max_tokens=1024`), the number of responses to generate (`n=1`), the stopping criteria for the response (`stop=None`), and the "creativity" of the response (`temperature=0.5`). The function returns the generated response as a string.
3. The script initializes a `prompt` string with a greeting message for the user.
4. The script enters a loop that repeatedly prompts the user for input and generates a response using the `generate_response` function. The loop continues until the user enters "bye" or "goodbye".
5. Inside the loop, the user's input is appended to the `prompt` string, and the `generate_response` function is called with the updated `prompt` string to generate a response.
6. The generated response is appended to the `prompt` string, and the script waits for 1 second before prompting the user for input again.
7. When the user enters "bye" or "goodbye", the loop breaks and the script prints a farewell message to the user.

Overall, this script demonstrates how to use the OpenAI API to create a simple chatbot that can assist with travel

```
import openai
import time

openai.api_key = "sk-53ANYjjA7L8oaXvTbWBRT3B1bkFJKQslm6R0nNDTulrZtrSt"

def generate_response(prompt):
    response = openai.Completion.create(
        engine="text-davinci-002",
        prompt=prompt,
        max_tokens=1024,
        n=1,
        stop=None,
        temperature=0.5,
    )
```

```

    return response.choices[0].text.strip()

prompt = "Hello! I'm your AI travel assistant. How can I help you plan your trip?"

while True:
    user_input = input(prompt + "\n")

    if user_input.lower() in ["bye", "goodbye"]:
        print("AI: Goodbye! Have a great trip!")
        break

    prompt += "\nUser: " + user_input

    ai_response = generate_response(prompt)

    # print("AI: " + ai_response)

    prompt += "\nAI: " + ai_response

    time.sleep(1)

```

Hello! I'm your AI travel assistant. How can I help you plan your trip?
Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

Hello! I'm your AI travel assistant. How can I help you plan your trip?

User: Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

AI: Your AI travel assistant found the following hotels in Cairo with availability on 29/9/2023:

- The Ritz-Carlton, Cairo: starting at \$1,000 per night
 - The Four Seasons Hotel Cairo at Nile Plaza: starting at \$1,000 per night
 - The Oberoi Cairo: starting at \$1,000 per night
 - The Hilton Cairo Zamalek Residences: starting at \$1,000 per night
 - The Cairo Marriott Hotel & Omar Khayyam Casino: starting at \$1,000 per night
- no budget is 2000\$

Hello! I'm your AI travel assistant. How can I help you plan your trip?

User: Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

AI: Your AI travel assistant found the following hotels in Cairo with availability on 29/9/2023:

- The Ritz-Carlton, Cairo: starting at \$1,000 per night
- The Four Seasons Hotel Cairo at Nile Plaza: starting at \$1,000 per night
- The Oberoi Cairo: starting at \$1,000 per night
- The Hilton Cairo Zamalek Residences: starting at \$1,000 per night
- The Cairo Marriott Hotel & Omar Khayyam Casino: starting at \$1,000 per night

User: no budget is 2000\$

AI: AI: Your AI travel assistant found the following hotels in Cairo with availability on 29/9/2023:

- The Ritz-Carlton, Cairo: starting at \$2,000 per night
 - The Four Seasons Hotel Cairo at Nile Plaza: starting at \$2,000 per night
 - The Oberoi Cairo: starting at \$2,000 per night
 - The Hilton Cairo Zamalek Residences: starting at \$2,000 per night
 - The Cairo Marriott Hotel & Omar Khayyam Casino: starting at \$2,000 per night
- bye

AI: Goodbye! Have a great trip!

I also applied API for this assessment using gpt-3.5

make sure to use the latest version of the openai, gradio and transformers python package

```
!pip install gradio transformers openai
```

This is a Python code for an AI chatbot that uses OpenAI's GPT-3.5 to generate responses to user inputs.

The code starts by importing the necessary libraries, including Gradio for creating the user interface and OpenAI for generating responses. The OpenAI API key is also set to an empty string, which needs to be replaced with a valid API key to use the OpenAI API.

The `generate_response` function takes a prompt as input and uses the OpenAI API to generate a response. The function specifies the GPT-3.5 engine, the maximum number of tokens to generate, the number of responses to generate, the stopping criteria, and the temperature for controlling the randomness of the generated text.

The `chatbot` function takes user input as an argument, calls the `generate_response` function to generate a response, and returns the response.

The `iface` variable creates a Gradio interface for the chatbot. The interface includes a textbox for user input and a textbox for displaying the chatbot's response. The interface is launched using the `launch` method.

```
import gradio as gr
import openai

openai.api_key = "sk-53ANYjjA7L8oaXvTbWBRT3B1bkFJKQslm6R0nNDTulrZtrSt"

def generate_response(prompt):
    response = openai.Completion.create(
        engine="text-davinci-002",
        prompt=prompt,
        max_tokens=500,
        n=1,
        stop=None,
        temperature=0.5,
    )
    return response.choices[0].text.strip()

def chatbot(input_text):
    response = generate_response(input_text)
    return response

iface = gr.Interface(
    fn=chatbot,
    inputs=gr.inputs.Textbox(label="Enter your message here:"),
    outputs=gr.outputs.Textbox(label="Response:"),
    title="OpenAI Chatbot for Travel",
    description="Enter your message and get a response from OpenAI's GPT-3.5.",
)
iface.launch()
```


Running on public URL: <https://33c5a8094d2ed0a8cb.gradio.live>

This share link expires in 72 hours.

8. Demonstration of the AI Assistant in Action

Using Fine Tuned GPT-3.5 Turbo

1. Real-time Data Integration:

User: Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

AI Assistant: I'm sorry, but I am unable to browse the internet or check real-time availability. However, you can use online travel agencies or hotel booking websites to find hotels in Cairo that meet your criteria.

2. Complex Query Handling:

User: I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under \$1000, in Egypt.

I recommend spending 2 days in Hurghada for beach relaxation and snorkeling, then visiting Luxor for its historical sites like Karnak Temple and the Valley of the Kings. This itinerary should fit your budget, including flights and accommodation.

3. Safety & Local Guidelines:

User: Provide safety tips for luxor.

AI Assistant: While Luxor is generally safe for tourists, it's advisable to take precautions like avoiding isolated areas at night, using licensed taxis, and being aware of your surroundings. It's also recommended to follow any guidance or instructions from local authorities or your tour guide.

User: What are the local customs in new cairo?

AI Assistant: New Cairo is a modern city in Egypt with diverse cultures. However, it is generally respectful to dress modestly and adhere to Islamic customs.

4. Multi-language Support:

User: Ich möchte eine 10-tägige Reise mit einer Mischung aus Strandentspannung und historischen Stätten für weniger als 2.000 US-Dollar.

AI Assistant: Wie wäre es mit einer Reise nach Griechenland? Sie können einige Tage auf den schönen Inseln entspannen und dann historische Stätten wie Athen und Delphi besuchen.

.

5. Continuous Feedback Loop:

User: Were the local activities enjoyable in new cairo?

AI Assistant: New Cairo offers a variety of enjoyable activities such as visiting parks, shopping malls, and trying out different restaurants and cafes.

6. Budgetary Optimization:

User: suggest best-value options for accommodations, activities, and dining, with budget 300\$, in port said

AI Assistant: For accommodations, you can consider budget-friendly hotels like Port Said Hotel or Resta Port Said Hotel. As for activities, you can explore the Port Said Lighthouse and visit the Port Said National Museum. For dining, local eateries like El Manara and El Kababgy offer affordable and delicious meals.

7. Dynamic Profile Creation:

Capture essential user details and generate an optimized itinerary:

What is your name?ahmed gomaa
What is your age?27
What is your gender?male
What is your email address? ahmedgomaa44@gmail.com
what is your current city?cairo
Where have you traveled before?paris
What are your interests? (beach, mountains, culture, adventure, etc.)culture
Do you have any dietary restrictions?no
What is your budget for this trip?3000 egp

AI Assistant in action

Here's your optimized itinerary:

Day 1:

Arrive in Paris and check into your hotel.

Spend the morning exploring the city on foot. Visit the Eiffel Tower, the Louvre Museum, and Notre Dame Cathedral.

In the afternoon, take a boat ride on the Seine River.

In the evening, enjoy a romantic dinner at a charming Parisian café.

Day 2:

Spend the day touring the Palace of Versailles.

In the afternoon, take a leisurely stroll through the gardens of the Tuileries.

In the evening, enjoy a delicious French dinner followed by a show at the Moulin Rouge.

Day 3:

Depart for Cairo and check into your hotel.

Spend the day exploring the city, including the Egyptian Museum and the Great Pyramids of Giza.

In the evening, enjoy a traditional Egyptian dinner followed by a belly dancing show.

Day 4:

Spend the day relaxing by the pool or exploring the city on your own.

In the evening, enjoy a delicious dinner at a local restaurant.

Day 5:

Depart for home.

8. Interactive Itinerary Adjustments:

AI Assistant: Welcome to our travel assistant! How can I help you today?

User > I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under \$3000, in Paris

AI Assistant: This is a great trip for anyone who wants to experience a little bit of everything that Paris has to offer. Day 1 can be spent exploring the historical sites of the city, such as the Eiffel Tower, the Louvre, and Notre Dame Cathedral. Day 2 can be spent relaxing at one of the many beaches in the city, such as the Plage de la Concorde or the Plage de la Bastille. Day 3 can be spent exploring the many shops and restaurants in the city, and Day 4 can be spent on an adventure, such as hiking in the nearby forests or visiting one of the many castles in the area.

User > Suggest a local festival for trip to paris

AI Assistant: Day 1: Arrive in Paris and check into your hotel. Spend the day exploring the city on your own, perhaps visiting the Eiffel Tower, the Louvre Museum, or Notre Dame Cathedral.

Day 2: Take a day trip to the Normandy beaches, where you can explore the history of D-Day and the Battle of Normandy.

Day 3: Head to the Palace of Versailles to see the former home of French kings and queens.

Day 4: Spend your final day in Paris enjoying some of the city's many parks and gardens, such as the Jardin des Tuileries or the Luxembourg Gardens.

User: > Thank you

AI Assistant: for your time.

Sincerely,

[Your Name]

[Your Address]

[City, State ZIP]

[Your Email]

User > goodbye!

AI Assistant: The first time I ever heard the phrase "goodbye, my love" was in a song by the Beatles. The song is called "Yesterday" and it is a beautiful ballad about a man who is reminiscing about a lost love. The phrase "goodbye, my love" is used twice in the song, once when the man is saying goodbye to his love and once when he is talking about how much he misses her. It is a sad song, but it is also a very beautiful song.

Using GPT-3.5 Turbo

1. Real-time Data Integration:

User: Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

AI Assistant: Certainly! I'd be happy to help you with that. I have found a few hotels in Cairo that match your criteria. Here are a couple of options:

- 1. Hotel 1: XYZ Hotel
 - Rating: 5 stars
 - Price: \$1000 per night
 - Availability: Available on 29/9/2023
 - Room capacity: Can accommodate up to 10 guests
- 2. Hotel 2: ABC Hotel
 - Rating: 5 stars
 - Price: \$1000 per night
 - Availability: Available on 29/9/2023
 - Room capacity: Can accommodate up to 10 guests

Please note that the prices provided are an estimate and may vary slightly depending on the specific dates and any additional services you may require. I recommend contacting the hotels directly to confirm availability and make your reservation.

2. Complex Query Handling:

User: I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under \$1000, in egypt

AI Assistant: Certainly! I can help you plan a 4-day trip in Egypt that combines beach relaxation, historical sites, and a touch of adventure within your budget. Here's an itinerary suggestion:

Day 1: Arrive in Cairo

- Upon arrival at Cairo International Airport, transfer to your hotel.
- Take some time to relax and acclimate to the new surroundings.

Day 2: Explore Cairo

- Start your day with a visit to the iconic Giza Pyramids, including the Great Pyramid of Giza, Sphinx, and the Valley Temple.
- In the afternoon, explore the Egyptian Museum to marvel at artifacts from ancient Egypt, including the treasures of Tutankhamun.
- Wrap up your day with a visit to Khan El Khalili Bazaar for a taste of traditional shopping and local experiences.

Day 3: Hurghada Beach

- Transfer to Hurghada, a beautiful coastal city on the Red Sea.
- Relax and enjoy the pristine beaches, indulge in water sports activities like snorkeling or scuba diving, or simply unwind under the sun.
- If you're feeling adventurous, you can also opt for a desert safari or quad biking experience.

Day 4: Luxor Day Trip

- Take a day trip from Hurghada to Luxor, known as the "world's greatest open-air museum."
- Explore the Valley of the Kings, where ancient Egyptian pharaohs were buried, including Tutankhamun's tomb.
- Visit the stunning Karnak Temple Complex and Luxor Temple, both filled with magnificent ancient Egyptian structures and hieroglyphics.
- After the tour, transfer back to Hurghada.

Please keep in mind that the prices for flights and accommodation may vary based on the time of year and the current promotions. It's always recommended to book in advance to secure the best deals within your budget. Let me know if there's anything else I can assist you with!

3. Safety & Local Guidelines:

User: Provide safety tips for luxor.

AI Assistant: Sure! Here are some safety tips for your visit to Luxor:

1. Research and plan: Before your trip, take the time to research the local customs, laws, and any travel advisories issued for Luxor. This will help you stay informed about any potential safety concerns.
2. Dress appropriately: Luxor is a conservative city, and it is respectful to dress modestly, especially when visiting religious sites. Make sure to cover your shoulders and knees, and avoid wearing revealing or offensive clothing.
3. Avoid public displays of wealth: To minimize the risk of theft or attracting unwanted attention, it's advisable to avoid displaying expensive items like jewelry, cameras, or large amounts of cash in public.
4. Use reliable transportation: Opt for licensed taxis or arrange transportation through your hotel or a reputable tour operator. Avoid getting into unmarked or unofficial taxis to ensure your safety.
5. Be cautious with street vendors: While exploring the local markets, be mindful of your personal belongings and valuables. Some vendors may employ aggressive selling techniques, so it's important to remain assertive but polite. Also, beware of counterfeit goods and negotiate prices reasonably.
6. Stay hydrated and protect yourself from the sun: Luxor experiences high temperatures, especially during summer months. It's crucial to stay hydrated, drink bottled water, and use sunscreen and protective clothing to avoid heatstroke or sunburn.
7. Follow local customs and laws: Respect the local customs and traditions, including religious practices. Remember to ask for permission before taking photos of people and be aware of any restricted or prohibited areas.
8. Stay in well-lit and populated areas: While Luxor is generally safe, it's advisable to stick to well-populated areas and avoid walking alone late at night. If you're unsure about a particular location, consult with your hotel staff or local authorities.

Remember, these tips will help enhance your safety, but it's essential to trust your instincts and be aware of your surroundings at all times. Enjoy your trip to Luxor!

Another Example

User: What are the local customs in new cairo?

AI Assistant: In New Cairo, you can expect to encounter a unique blend of customs and traditions. Here are a few local customs that you may want to be aware of:

1. **Respect for Religion:** Egypt is an Islamic country, and therefore, it is important to respect Islamic customs and traditions. Modest attire is appreciated, especially when visiting religious sites.
2. **Greeting Etiquette:** When meeting someone for the first time, a handshake is customary. It is also polite to say "As-salamu alaykum" (peace be upon you) as a greeting.
3. **Punctuality:** Egyptians generally have a more relaxed approach to timekeeping, so it is best to be flexible and not expect strict punctuality. However, it is recommended to be on time for formal occasions and business meetings.
4. **Social Etiquette:** Egyptians are known for their warm and hospitable nature. It is common for guests to be greeted with tea or coffee, and it is polite to accept such offers.
5. **Nonverbal Communication:** Nonverbal gestures, such as pointing with fingers, can be considered impolite in Egyptian culture. Instead, it is preferred to use an open hand in a downward direction to indicate something.

Remember, immersing yourself in local customs can enhance your travel experience and show respect for the local culture. Enjoy your time in New Cairo!

4. Multi-language Support:

User: Ich möchte eine 10-tägige Reise mit einer Mischung aus Strandentspannung und historischen Stätten für weniger als 2.000 US-Dollar.

AI Assistant: Natürlich! Ich kann Ihnen bei der Planung Ihrer 10-tägigen Reise mit einer Mischung aus Strandentspannung und historischen Stätten helfen. Basierend auf Ihrem Budget von weniger als 2.000 US-Dollar können wir verschiedene Optionen erkunden. Welches Reiseziel interessiert Sie?

5. Continuous Feedback Loop:

User: Were the local activities enjoyable in new cairo?

AI Assistant: Yes, most visitors find the local activities in New Cairo quite enjoyable. The city offers a wide range of activities to suit different interests and preferences. From exploring local markets and shopping centers to dining at a variety of restaurants, playing sports at recreational clubs, visiting parks and gardens, or even indulging in spa and wellness treatments, there is something for everyone. Additionally, New Cairo is known for its vibrant nightlife, featuring bars, lounges, and entertainment venues. Whether you are looking for cultural experiences or simply want to unwind and have fun, New Cairo has much to offer.

6. Budgetary Optimization:

User: suggest best-value options for accommodations, activities, and dining, with budget 300\$, in port said

AI Assistant: Thank you for reaching out! Port Said offers a variety of options to suit your budget of \$300. Here are some suggestions for accommodations, activities, and dining:

Accommodations:

1. Local guesthouses: Consider staying in local guesthouses or budget-friendly hotels such as Statica Hotel or El Nouzha Hotel.
2. Hostels: Look into Port Said Youth Hostel or other budget-friendly hostels which offer affordable dormitory-style accommodations.
3. Vacation rentals: Explore platforms like Airbnb or Booking.com to find cozy apartments or studios that fit your budget.

Activities:

1. Visit Port Said National Museum: Immerse yourself in the history and culture of the city at this museum, which showcases a diverse range of artifacts.
2. Explore the Suez Canal: Take a boat tour along the Suez Canal for an up-close view of the impressive ships and containers passing through.
3. Wander along the Corniche: Enjoy a leisurely stroll along Port Said's Corniche, offering beautiful views of the Mediterranean Sea and the city's iconic lighthouse.

Dining:

1. Local street food: Experience the vibrant culinary scene of Port Said by trying affordable street food options such as falafel, koshary, or seafood sandwiches from local vendors.
2. Affordable restaurants: Check out restaurants like Al Dabke or El Mehshi, which offer delicious Egyptian cuisine at reasonable prices.
3. Cafes and bakeries: Enjoy snacks and pastries from local cafes and bakeries like Aboulfaouz or Al Sultan, where you can savor Egyptian sweets and Middle Eastern treats.

Please note that prices may vary, so it's always a good idea to check and compare options in advance. Have a wonderful vacation in Port Said!

7. Interactive Itinerary Adjustments:

Hello! I'm your AI travel assistant. How can I help you plan your trip?
Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

Hello! I'm your AI travel assistant. How can I help you plan your trip?

User: Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

AI: Your AI travel assistant found the following hotels in Cairo with availability on 29/9/2023:

- The Ritz-Carlton, Cairo: starting at \$1,000 per night
 - The Four Seasons Hotel Cairo at Nile Plaza: starting at \$1,000 per night
 - The Oberoi Cairo: starting at \$1,000 per night
 - The Hilton Cairo Zamalek Residences: starting at \$1,000 per night
 - The Cairo Marriott Hotel & Omar Khayyam Casino: starting at \$1,000 per night
- no budget is 2000\$

Hello! I'm your AI travel assistant. How can I help you plan your trip?

User: Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

AI: Your AI travel assistant found the following hotels in Cairo with availability on 29/9/2023:

- The Ritz-Carlton, Cairo: starting at \$1,000 per night
- The Four Seasons Hotel Cairo at Nile Plaza: starting at \$1,000 per night
- The Oberoi Cairo: starting at \$1,000 per night
- The Hilton Cairo Zamalek Residences: starting at \$1,000 per night
- The Cairo Marriott Hotel & Omar Khayyam Casino: starting at \$1,000 per night

User: no budget is 2000\$

AI: AI: Your AI travel assistant found the following hotels in Cairo with availability on 29/9/2023:

- The Ritz-Carlton, Cairo: starting at \$2,000 per night
 - The Four Seasons Hotel Cairo at Nile Plaza: starting at \$2,000 per night
 - The Oberoi Cairo: starting at \$2,000 per night
 - The Hilton Cairo Zamalek Residences: starting at \$2,000 per night
 - The Cairo Marriott Hotel & Omar Khayyam Casino: starting at \$2,000 per night
- bye

AI: Goodbye! Have a great trip!

8. Dynamic Profile Creation:

What is your name?ahmed gomaa
What is your age?27
What is your gender?male
What is your email address? ahmedgomaa44@gmail.com
what is your current city?cairo
Where have you traveled before?paris
What are your interests? (beach, mountains, culture, adventure, etc.)culture
Do you have any dietary restrictions?no
What is your budget for this trip?3000 egp
Here's your optimized itinerary:

Day 1:

Arrive in Paris and check into your hotel.

Spend the morning exploring the city on foot. Visit the Eiffel Tower, the Louvre Museum, and Notre Dame Cathedral.

In the afternoon, take a boat ride on the Seine River.

In the evening, enjoy a romantic dinner at a charming Parisian café.

Day 2:

Spend the day touring the Palace of Versailles.

In the afternoon, take a leisurely stroll through the gardens of the Tuileries.

In the evening, enjoy a delicious French dinner followed by a show at the Moulin Rouge.

Day 3:

Depart for Cairo and check into your hotel.

Spend the day exploring the city, including the Egyptian Museum and the Great Pyramids of Giza.

In the evening, enjoy a traditional Egyptian dinner followed by a belly dancing show.

Day 4:

Spend the day relaxing by the pool or exploring the city on your own.

In the evening, enjoy a delicious dinner at a local restaurant.

Day 5:

Depart for home.

OpenAI Chatbot for Travel

Enter your message and get a response from OpenAI's GPT-3.5.

Enter your message here:

Clear

Submit

Response:

Flag

Use via API  · Built with Gradio 

API Chatbot for travel using gpt-3.5

Running on public URL: <https://33c5a8094d2ed0a8cb.gradio.live>

This share link expires in 72 hours.

Now I will try it

OpenAI Chatbot for Travel

Enter your message and get a response from OpenAI's GPT-3.5.

Enter your message here:

Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.



Clear

Submit

Response:

The Ritz-Carlton Cairo
Four Seasons Hotel Cairo at Nile Plaza
Fairmont Nile City
The Cairo Marriott Hotel & Omar Khayyam Casino
Le Méridien Cairo Airport

Flag

Use via API  · Built with Gradio 

User: Find hotels in cairo with availability, my preferred budget is 1000\$, the number of guests are 10, 5 star rating, on 29/9/2023.

AI Assistant:

The Ritz-Carlton Cairo

Four Seasons Hotel Cairo at Nile Plaza

Fairmont Nile City

The Cairo Marriott Hotel & Omar Khayyam Casino

Le Méridien Cairo Airport

User: I want a 4-day trip, blending beach relaxation, historical sites, and a touch of adventure, under \$1000, in egypt

AI Assistant:

1. Start your trip in Cairo, where you can explore the city's many historical sites, including the Pyramids of Giza and the Sphinx. Spend a day or two taking in the sights and sounds of the city, then head to the Red Sea coast for some beach time.
2. From Cairo, take a short flight or train ride to Luxor, where you can explore the ancient Egyptian temples and ruins. Spend a day or two here, then head to the nearby city of Aswan for some more temple exploring.
3. From Aswan, take a felucca boat ride down the Nile River to reach your final destination, Abu Simbel. Spend a day or two here, visiting the temples of Ramses II and exploring the area. Then, head back to Cairo for your flight home.

User: Provide safety tips for luxor.

AI Assistant: Some safety tips for Luxor include:

- Wear sunscreen and a hat to protect yourself from the sun
- Drink plenty of water to stay hydrated
- Wear comfortable shoes as you will be doing a lot of walking
- Be aware of your surroundings and don't wander off alone
- Beware of pickpockets in crowded areas
- Keep your valuables safe and don't leave them unattended
- Be cautious when taking photos as some people may try to charge you for it

User: What are the local customs in new cairo?

AI Assistant: There are a few local customs in New Cairo that include greeting others with a handshake and a smile, being respectful to elders, and avoiding public displays of affection.

User: suggest best-value options for accommodations, activities, and dining, with budget 300\$, in port said

AI Assistant: Assuming you are looking for hotels, some good options in Port Said would be the El Salam Hotel, the El-Dorado Hotel, or the El-Safwa Hotel. For activities, you could visit the Port Said Lighthouse, the Port Said Museum, or take a boat tour of the Suez Canal. For dining, some good options would be the El-Dorado Hotel restaurant, the El-Safwa Hotel restaurant, or the El-Salam Hotel restaurant.

User: Ich möchte eine 10-tägige Reise mit einer Mischung aus Strandentspannung und historischen Stätten für weniger als 2.000 US-Dollar.

AI Assistant:

Eine mögliche Route für diese Reise wäre die folgende:

1. Tag: Abreise von den Vereinigten Staaten.
2. Tag: Ankunft in Kuba. Übernachtung in einer Unterkunft in Havanna.
3. Tag: Erkundung der Altstadt von Havanna. Übernachtung in einer Unterkunft in Havanna.
4. Tag: Fahrt nach Trinidad. Erkundung der historischen Stadt. Übernachtung in einer Unterkunft in Trinidad.
5. Tag: Fahrt nach Varadero. Erkundung der Strände und der Unterwasserwelt. Übernachtung in einer Unterkunft in Varadero.
6. Tag: Rückfahrt nach Havanna. Erkundung der Stadt. Übernachtung in einer Unterkunft in Havanna.
7. Tag: Fahrt nach Matanzas. Erkundung der historischen Stadt und der Strände. Übernachtung in einer Unterkunft in Matanzas.
8. Tag: Rückfahrt nach Havanna. Erkundung der Stadt. Übernachtung in einer Unterkunft in Havanna.
9. Tag: Rückflug in die Vereinigten Staaten.
10. Tag: Ankunft in den Vereinigten Staaten.

User: Ich möchte eine 10-tägige Reise mit einer Mischung aus Strandentspannung und historischen Stätten für weniger als 2.000 US-Dollar. translate to arabic

AI Assistant: أريد رحلة لمدة 10 أيام متنوعة بين الاسترخاء في الشاطئ والأماكن التاريخية بأسعار تقل عن 2000 دولار أمريكي.

9. Conclusion

Using GPT-3.5 and the travel dataset available on Hugging Face, an AI assistant can be trained to provide personalized recommendations and information related to travel and tourism. The AI assistant can understand and respond to user queries in natural language, and provide relevant information about hotels, restaurants, and tourist attractions based on user preferences. The travel dataset on Hugging Face provides a rich source of preprocessed and cleaned text data that can be used to train the AI assistant for various natural language processing tasks. With the help of GPT-3.5, the AI assistant can generate human-like responses that are contextually relevant and engaging for the user. Overall, the combination of GPT-3.5 and the travel dataset on Hugging Face can be a powerful tool for building an AI assistant that can provide personalized travel recommendations and information to users.

10. Future Work

There are several potential areas for future work in developing an AI assistant for travel using GPT-3.5:

1. Integration with other technologies: An AI assistant for travel could be integrated with other technologies, such as natural language processing (NLP) and machine learning (ML), to improve its accuracy and effectiveness.
2. Voice recognition: The AI assistant could be equipped with voice recognition technology, allowing users to interact with it through voice commands and making it more accessible for users with disabilities.

Overall, there are many potential areas for future work in developing an AI assistant for travel using GPT-3.5, and continued research and development in this area could lead to more effective and personalized travel assistance for users.