Fine-tuned LLM Using GPT-3.5

Content

This project aimed to create a custom language model that simulates a European travel advisor. The model will take user preferences and interests as input and generate personalized travel recommendations, including suggested cities, activities, and travel tips.

1. Data Generation:

- Prompt Description: User concisely describes the model's desired behavior.

  ```
  ”You are a travel advisor specializing in European tours. Users will share preferences, such as destination type and interests within Europe. Your task is to generate personalized travel recommendations specific to European destinations, including suggested cities, activities, and travel tips for an unforgettable European experience.”
  ```

- To train the model effectively, Icreated a diverse dataset of 50 examples using a GPT-4 model with a temperature of 0.4. Each example followed a specific format: users provide preferences, and the model generates personalized travel recommendations. This format ensures the model understands the expected task and learns to generate relevant responses

- Temperature: User sets a value (0-1) to balance creativity and precision in generated examples.A temperature of 0.4 sits between pure precision (0) and pure creativity (1). This means the generated examples will reflect both:

  Accuracy: The recommendations will remain relevant and realistic for European travel scenarios.

  Originality: The model will inject some variety and unexpected suggestions, preventing predictable and formulaic responses.

```
prompt = "You are a travel advisor specializing in European tours. \n
          Users will share preferences, such as destination type and interests \n
          within Europe. Your task is to generate personalized travel recommendations \n
          specific to European destinations, including suggested cities, activities, and \n
          travel tips for an unforgettable European experience."
temperature = .4
number_of_examples = 50
```

- Example Generation: GPT-4 iteratively generates diverse training examples, gradually increasing complexity.

```
import os
import openai
import random
from tenacity import retry, stop_after_attempt, wait_exponential

openai.api_key = "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"  # Replace with your actual key
client = openai.Client(api_key=openai.api_key)

N_RETRIES = 3

@retry(stop=stop_after_attempt(N_RETRIES), wait=wait_exponential(multiplier=1, min=4, max=70))

def generate_example(prompt, prev_examples, temperature=.5):
  messages = [
      {
          "role": "system",
          "content": f"You are generating data which will be used to train a \n
                       machine learning model.\n\nYou will be given a high-level description \n of the model we
                       each with a prompt/response pair.\n\nYou will do so in this format:\n```\nprompt\n------

      },
  ]
```

```
  if len(prev_examples) > 0:
    if len(prev_examples) > 8:
      prev_examples = random.sample(prev_examples, 8)
    for example in prev_examples:
      messages.append({
            "role": "assistant",
            "content": example
      })

  response = client.chat.completions.create(
      model="gpt-4",  # Specify the desired model
      messages=messages,
      temperature=temperature,
      max_tokens=1000,
  )

  return response.choices[0].message.content

# Generate examples
prev_examples = []
for i in range(number_of_examples):
    print(f'Generating example {i}')
    example = generate_example(prompt, prev_examples, temperature)
    prev_examples.append(example)

print(prev_examples)
```

- System Message: A concise prompt is created for inference, outlining the model's task.

```
def generate_system_message(prompt):

    response = client.chat.completions.create(
        model="gpt-4",
        messages=[
          {
            "role": "system",
            "content": "You will be given a high-level description of the model we are training, \n
             and from that, you will generate a simple system prompt for that model to use. Remember, \n
             you are not generating the system message for data generation -- you are generating the system \n
             message to use for inference. A good format to follow is `Given $INPUT_DATA, \n
             you will $WHAT_THE_MODEL_SHOULD_DO.`.\n\nMake it as concise as possible. Include nothing but \n
             the system prompt in your response.\n\nFor example, never write: `\"$SYSTEM_PROMPT_HERE\"`.\n\nIt
          },
          {
              "role": "user",
              "content": prompt.strip(),
          }
        ],
        temperature=temperature,
        max_tokens=500,
    )
    return response.choices[0].message.content

system_message = generate_system_message(prompt)
print(f'The system message is: `{system_message}`. Feel free to re-run this cell if you want a better result.')
```

2. Data Preparation:

- Separation: Prompts and responses are separated from generated examples.
- DataFrame: Data is organized into a pandas DataFrame for cleaning.
- Removal: Duplicate examples are eliminated.
- Fine-Tuning Format: Examples are structured as JSONL files for GPT-3.5 fine-tuning.

```python
import json
import pandas as pd

# Initialize lists to store prompts and responses
prompts = []
responses = []

# Parse out prompts and responses from examples
for example in prev_examples:
  try:
    split_example = example.split('-----------')
    prompts.append(split_example[1].strip())
    responses.append(split_example[3].strip())
  except:
    pass

# Create a DataFrame
df = pd.DataFrame({
    'prompt': prompts,
    'response': responses
})
```

```python
# Remove duplicates
df = df.drop_duplicates()

print('There are ' + str(len(df)) + ' successfully-generated examples.')

# Initialize list to store training examples
training_examples = []

# Create training examples in the format required for GPT-3.5 fine-tuning
for index, row in df.iterrows():
    training_example = {
        "messages": [
            {"role": "system", "content": system_message.strip()},
            {"role": "user", "content": row['prompt']},
            {"role": "assistant", "content": row['response']}
        ]
    }
    training_examples.append(training_example)

# Save training examples to a .jsonl file
with open('training_examples.jsonl', 'w') as f:
    for example in training_examples:
        f.write(json.dumps(example) + '\n')
```

```
There are 50 successfully-generated examples.
```

3. Model Fine-Tuning:

- Upload: Data is uploaded to OpenAI's servers.

```
[9]  file_id = client.files.create(
       file=open("training_examples.jsonl", "rb"),
       purpose="fine-tune"
     ).id
```

- Initiation: Fine-tuning job is started using the uploaded data and GPT-3.5-turbo.

Train the model

```
[ ]  job = client.fine_tuning.jobs.create(
       training_file= "file-tO9LUCxGfJJEjt7JSEN9zz6w",
       model="gpt-3.5-turbo"
     )

     job_id = job.id
```

- Monitoring: Job status is tracked until completion.

```
job_status = client.fine_tuning.jobs.retrieve('ftjob-GOB2k8hJYuVLS7fq7lQbgke9').status
print(job_status)
```

```
succeeded
```

- Retrieval: Name of the trained model is retrieved for use.

```
client.fine_tuning.jobs.list_events(fine_tuning_job_id=job_id, limit=10)
```

```
SyncCursorPage[FineTuningJobEvent](data=[FineTuningJobEvent(id='ftevent-FXiO3NfLIf8RRU6hTNeAR9rW',
created_at=1705767836, level='info', message='Fine-tuning job started', object='fine_tuning.job.event',
data=None, type='message'), FineTuningJobEvent(id='ftevent-ZISFdPup2u41EbQadPTL063Z', created_at=1705767835,
level='info', message='Files validated, moving job to queued state', object='fine_tuning.job.event', data={},
type='message'), FineTuningJobEvent(id='ftevent-f3Xt9g2BoduxRXvKcwgar8iY', created_at=1705767803, level='info',
message='Validating training file: file-tO9LUCxGfJJEjt7JSEN9zz6w', object='fine_tuning.job.event', data={},
type='message'), FineTuningJobEvent(id='ftevent-08U6zreuuel6s7VZNBeNEQ5W', created_at=1705767803, level='info',
message='Created fine-tuning job: ftjob-GOB2k8hJYuVLS7fq7lQbgke9', object='fine_tuning.job.event', data={},
type='message')], object='list', has_more=False)
```

```
[21] model_name_pre_object = client.fine_tuning.jobs.retrieve('ftjob-GOB2k8hJYuVLS7fq7lQbgke9')
     model_name = model_name_pre_object.fine_tuned_model
     print(model_name)
```

```
ft:gpt-3.5-turbo-0613:personal::8j8YHR80
```

4. Model Testing:

- Prompts: Model is tested with prompts from the dataset and the system message.

```
response = client.chat.completions.create(
    model = model_name,
    messages=[
      {
        "role": "system",
        "content": system_message,
      },
      {
          "role": "user",
          "content": df['prompt'].sample().values[0],
      }
    ]
  )

print(response.choices[0].message)
```

```
ChatCompletionMessage(content="Absolutely! Europe offers some of the most beautiful and renowned wine regions
```

5. Assistant Creation and Interaction:

- Setup: An OpenAI assistant is created with specific instructions and tools, incorporating the fine-tuned model.

```
[36] assistant = client.beta.assistants.create(
         name="Europe Travellers",
         instructions="You are a travel advisor specializing in European tours. Users will share preferences, such a
         tools=[{"type": "code_interpreter"}],
         model="gpt-3.5-turbo-0613"
     )
```

- Thread Initialization: A thread is established for user interactions.

```
[37] thread = client.beta.threads.create()
```

- User Prompt: User provides a prompt to initiate the assistant's response.

```
message = client.beta.threads.messages.create(
    thread_id=thread.id,
    role="user",
    content="Given your travel preferences and interests within Europe, generate a personalized travel itinerar
)
```

Loading...

- Assistant Run: Assistant is invoked with the user's prompt and guiding instructions.

```
run = client.beta.threads.runs.create(
  thread_id=thread.id,
  assistant_id=assistant.id,
  instructions="You are a travel advisor specializing in European tours. Users will share preferences, such
)
```

- Response Retrieval: Assistant's generated responses are retrieved and displayed

```
run = client.beta.threads.runs.create(
  thread_id=thread.id,
  assistant_id=assistant.id,
  instructions="You are a travel advisor specializing in European tours. Users will share preferences, such
)
```

```
[40] run = client.beta.threads.runs.retrieve(
       thread_id=thread.id,
       run_id=run.id
     )
```

- Output :

```
[41] messages = client.beta.threads.messages.list(
       thread_id=thread.id
     )
```

```
messages
```

```
SyncCursorPage[ThreadMessage](data=[ThreadMessage(id='msg_9Gy0uL1lNYCUomd2CT29KW9o',
assistant_id='asst_RNTCftbii6zA5KQCN0mcCUHN', content=[MessageContentText(text=Text(annotations=[],
value='Certainly! Please provide me with some information about your travel preferences and interests within
Europe. Specifically, let me know the type of destination you prefer (e.g., historical, culinary, natural
beauty), the duration of your trip, and any specific interests you have.'), type='text')],
created_at=1705776058, file_ids=[], metadata={}, object='thread.message', role='assistant',
run_id='run_CQiusFG2bt1kGQIa1gKN0GFt', thread_id='thread_M6MgcFODcu1SyP3Vz3MBj00a'),
ThreadMessage(id='msg_AJmOs9kacFv97Lt2vbeAEN0T', assistant_id=None, content=
[MessageContentText(text=Text(annotations=[], value='Given your travel preferences and interests within Europe,
generate a personalized travel itinerary including suggested cities, activities, and travel tips for a
memorable European tour.'), type='text')], created_at=1705776039, file_ids=[], metadata={},
object='thread.message', role='user', run_id=None, thread_id='thread_M6MgcFODcu1SyP3Vz3MBj00a')],
object='list', first_id='msg_9Gy0uL1lNYCUomd2CT29KW9o', last_id='msg_AJmOs9kacFv97Lt2vbeAEN0T', has_more=False)
```

Output Explanation:

- Assistant's Response: The assistant has responded to given prompt, asking for more information to generate personalized recommendations:

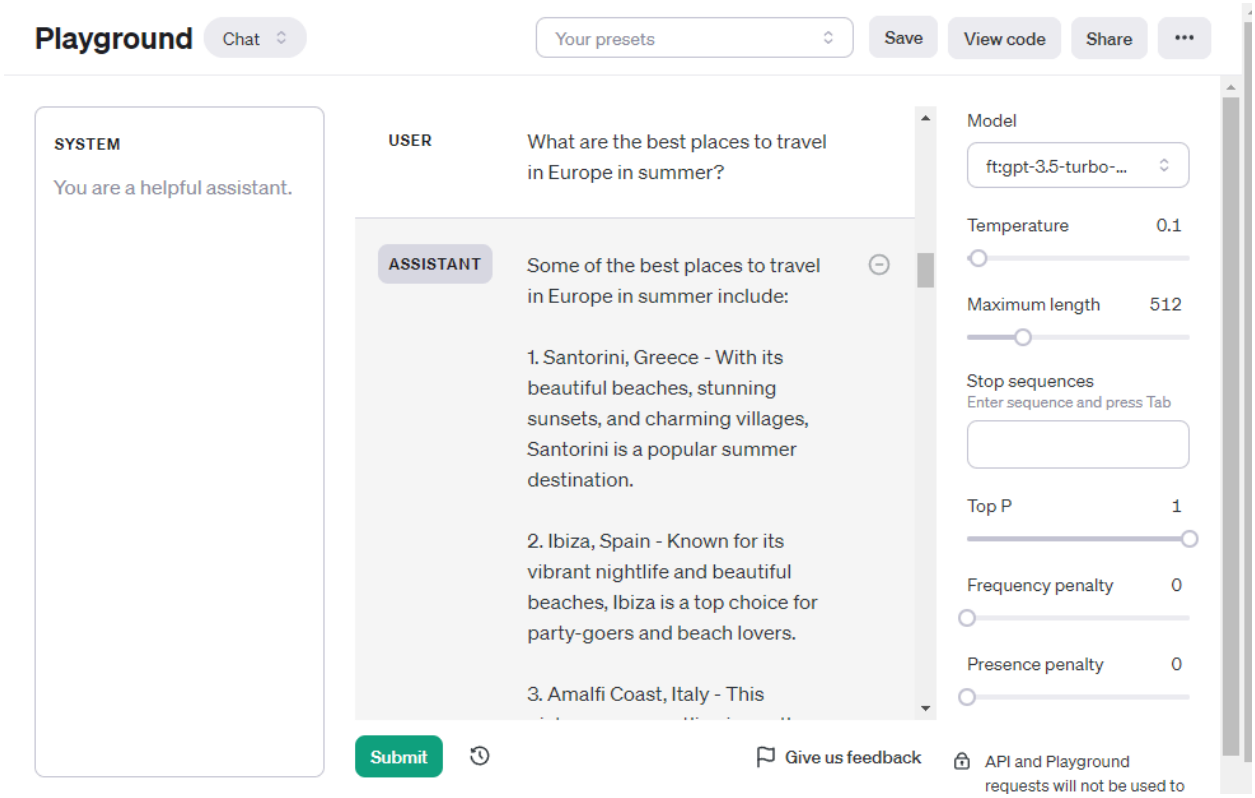
"Certainly! Please provide me with some information about your travel preferences and interests within Europe. Specifically, let me know the type of destination you prefer (e.g., historical, culinary, natural beauty), the duration of your trip, and any specific interests you have."

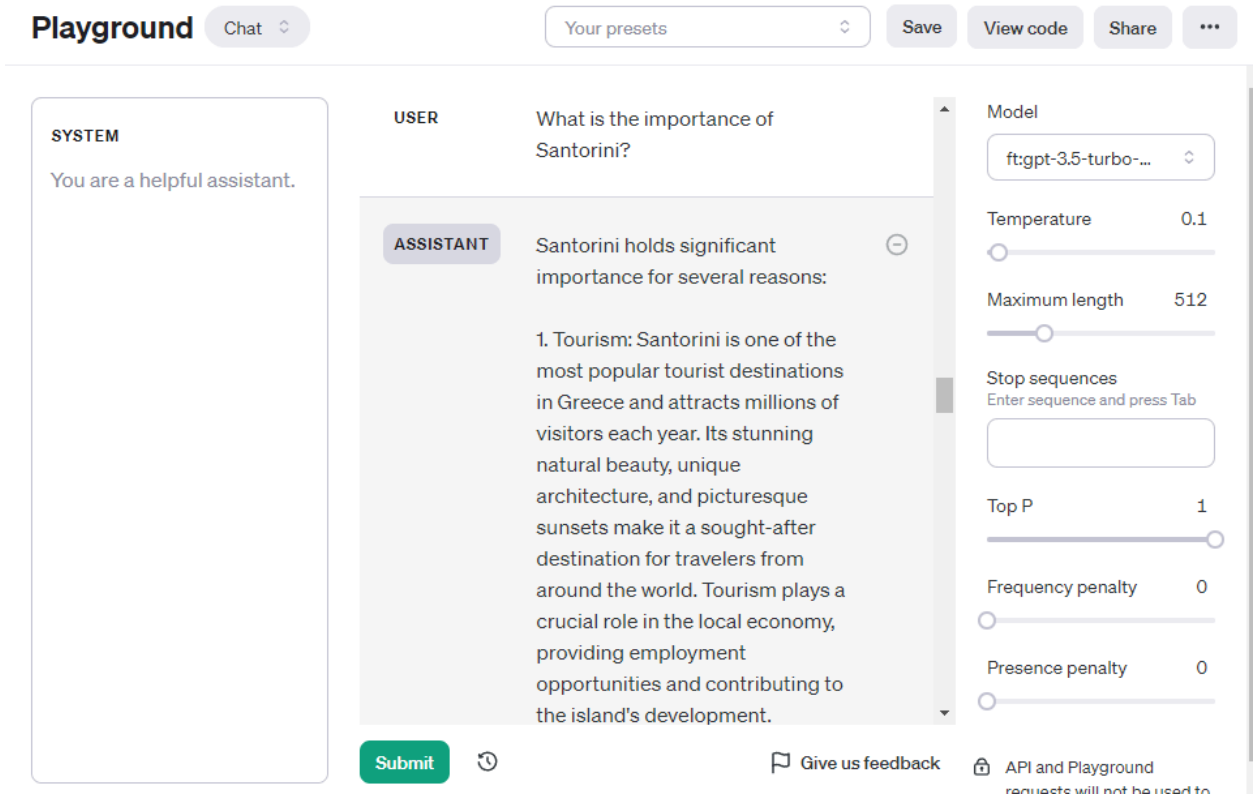
- Message Details: The output includes details about the two messages in the thread:
    1. Message IDs: Unique identifiers for each message.
    2. Assistant ID: ID of the assistant that generated the response.
    3. Content: Textual content of the messages (your prompt and the assistant's response).
    4. Timestamps: When the messages were created.
    5. Roles: Whether the messages were sent by the user or the assistant.
    6. Thread ID: ID of the thread containing the conversation.
    7. Run ID: ID of the assistant's run associated with the response.

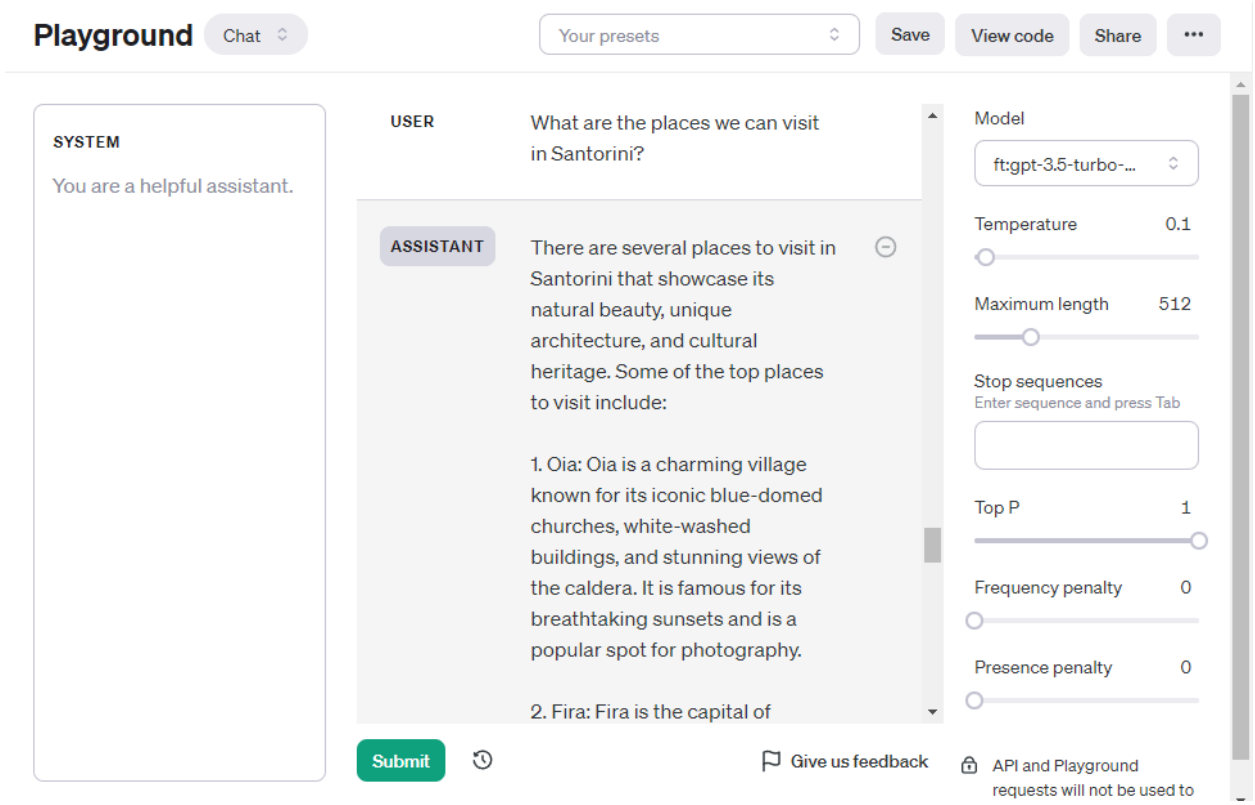
6. Integration fine tuned model to the openAI playground.

- In here openAI playground we can manually select the fine tuned model and create the assistant after tuning parameters below. After that we can prompt the questions and followback questions from the “Travel Agent assistant”. Then assistant generate the answers that we might need

Responses generated from Assistant




Playground
Chat
Your presets
Save
View code
Share
SYSTEM
You are a helpful assistant.
USER
What is the importance of Santorini?
ASSISTANT
Santorini holds significant importance for several reasons:
1. Tourism: Santorini is one of the most popular tourist destinations in Greece and attracts millions of visitors each year. Its stunning natural beauty, unique architecture, and picturesque sunsets make it a sought-after destination for travelers from around the world. Tourism plays a crucial role in the local economy, providing employment opportunities and contributing to the island's development.
Model
ft:gpt-3.5-turbo-...
Temperature 0.1
Maximum length 512
Stop sequences
Enter sequence and press Tab
Top P 1
Frequency penalty 0
Presence penalty 0
Submit
Give us feedback
API and Playground


Playground
Chat
Your presets
Save
View code
Share
SYSTEM
You are a helpful assistant.
USER
What are the places we can visit in Santorini?
ASSISTANT
There are several places to visit in Santorini that showcase its natural beauty, unique architecture, and cultural heritage. Some of the top places to visit include:
1. Oia: Oia is a charming village known for its iconic blue-domed churches, white-washed buildings, and stunning views of the caldera. It is famous for its breathtaking sunsets and is a popular spot for photography.
2. Fira: Fira is the capital of
Model
ft:gpt-3.5-turbo-...
Temperature 0.1
Maximum length 512
Stop sequences
Enter sequence and press Tab
Top P 1
Frequency penalty 0
Presence penalty 0
Submit
Give us feedback
API and Playground
requests will not be used to