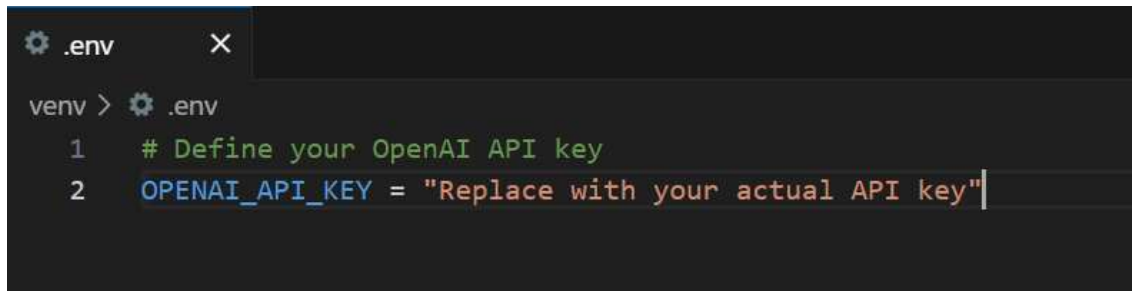


Fine-Tuning GPT-3 with custom dataset from JSON file

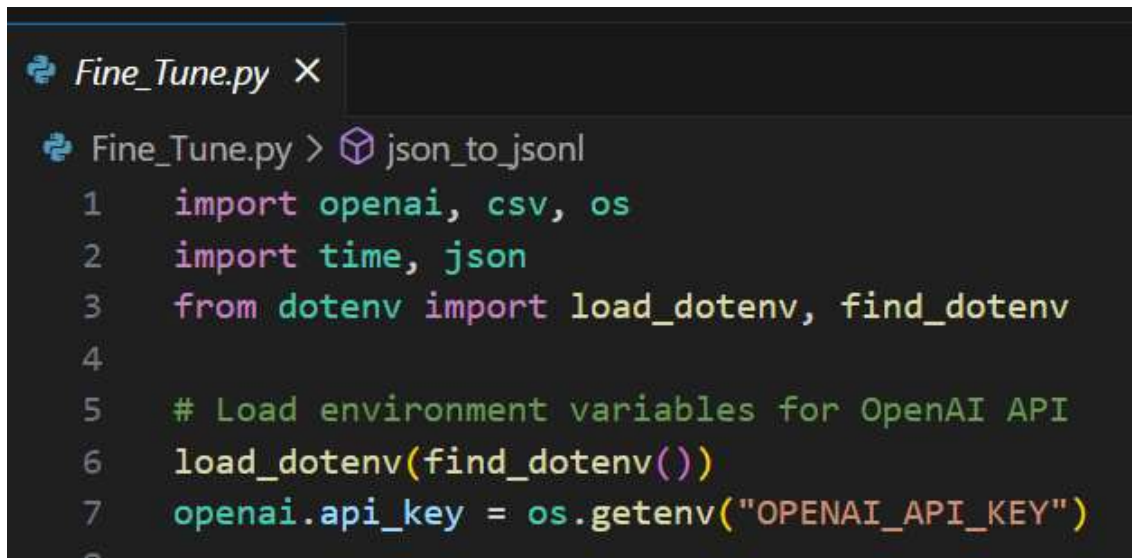
Set up environment and install required packages:



A screenshot of a code editor window titled ".env" with a close button (X). The editor shows a virtual environment prompt "venv >" followed by two lines of code: a comment "# Define your OpenAI API key" and an assignment "OPENAI_API_KEY = 'Replace with your actual API key'".

```
.env X
venv > .env
1 # Define your OpenAI API key
2 OPENAI_API_KEY = "Replace with your actual API key"
```

And load the key using load_dotenv and find_dotenv:



A screenshot of a code editor window titled "Fine_Tune.py" with a close button (X). The editor shows a Jupyter Notebook prompt "Fine_Tune.py >" followed by a code cell icon and the text "json_to_jsonl". Below this, there is Python code that imports openai, csv, os, time, and json, then imports load_dotenv and find_dotenv from the dotenv module. It then loads the environment variables and sets the openai.api_key to the value of the OPENAI_API_KEY environment variable.

```
Fine_Tune.py X
Fine_Tune.py > json_to_jsonl
1 import openai, csv, os
2 import time, json
3 from dotenv import load_dotenv, find_dotenv
4
5 # Load environment variables for OpenAI API
6 load_dotenv(find_dotenv())
7 openai.api_key = os.getenv("OPENAI_API_KEY")
8
```

Step 1: Use a Json Data File compatible with your base model.

```
Fine_Tune.py  {} data.json x
{} data.json > {} 1 > [ ] messages
1  [
2
3      {
4          "messages": [
5              {"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."},
6              {"role": "user", "content": "What's the capital of France?"},
7              {"role": "assistant", "content": "Paris, as if everyone doesn't know that already."}
8          ],
9      },
10     {
11         "messages": [
12             {"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."},
13             {"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
14             {"role": "assistant", "content": "Oh, just some guy named William Shakespeare. Ever heard of him?"}
15         ],
16     },
17     {
18         "messages": [
19             {"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."},
20             {"role": "user", "content": "How far is the Moon from Earth?"},
21             {"role": "assistant", "content": "Around 384,400 kilometers. Give or take a few, like that really matters."}
22         ],
23     },
24     {
25         "messages": [
26             {"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."},
27             {"role": "user", "content": "What is the largest planet in our solar system?"},
28             {"role": "assistant", "content": "Oh, it's just Jupiter. You know, the one that's so big you can't miss it."}
29         ],
30     },
31 ]
```

Step 2: Prepare the Data in JSONL file – Convert the json file to jsonl file using python code.

```
Fine_Tune.py x
Fine_Tune.py > save_metrics_to_csv
9  # Function to convert JSON to JSONL format
10 def json_to_jsonl(input_json_path, output_jsonl_path):
11     """
12     Converts a JSON file to JSONL format by writing each object in the JSON array to a new line in the output JS
13     """
14     try:
15         # Open and load the JSON file
16         with open(input_json_path, 'r') as json_file:
17             data = json.load(json_file)
18
19         # Write the data to the jsonl file, each item on a new line
20         with open(output_jsonl_path, 'w') as jsonl_file:
21             for item in data:
22                 jsonl_file.write(json.dumps(item) + '\n')
23
24         print(f"Conversion successful! {input_json_path} has been converted to {output_jsonl_path}")
25     except Exception as e:
26         print(f"Error during conversion: {e}")
27
28 # Example usage of the conversion function
29 input_json_path = "data.json" # Path to your input JSON file
30 output_jsonl_path = "data_prepared.jsonl" # Path to save the output JSONL file
31 json_to_jsonl(input_json_path, output_jsonl_path)
32
```

```

Fine_Tune.py  data_prepared.jsonl X
data_prepared.jsonl
1  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What's the capital of France?"}, {"role": "assistant", "co
2  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "Who wrote 'Romeo and Juliet?'}, {"role": "assistant", "co
3  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "How far is the Moon from Earth?"}, {"role": "assistant", "
4  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What is the largest planet in our solar system?"}, {"role"
5  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What is the boiling point of water?"}, {"role": "assistant
6  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "Who is the president of the United States?"}, {"role": "as
7  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What is the speed of light?"}, {"role": "assistant", "cont
8  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What is the capital of Australia?"}, {"role": "assistant",
9  {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What does DNA stand for?"}, {"role": "assistant", "content
10 {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "How many continents are there?"}, {"role": "assistant", "c
11 {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What is the tallest mountain in the world?"}, {"role": "as
12

```

Step 3: Use this JSONL file as training file for creating a fine-tuning model.

```
# Function to upload the training file to OpenAI
def upload_file(file_path):
    """
    Uploads the training file to OpenAI's server for fine-tuning.
    Returns the file ID if the upload is successful, else returns None.
    """
    try:
        response = openai.files.create(file=open(file_path, "rb"), purpose="fine-tune")
        print(f"File uploaded: {response.id}")
        return response.id
    except Exception as e:
        print(f"Error uploading file: {e}")
        return None

# Function to start the fine-tuning job
def start_fine_tuning(training_file_id, model="gpt-3.5-turbo-1106"):
    """
    Starts a fine-tuning job with the specified training file ID and model.
    Returns the job ID if the job starts successfully, else returns None.
    """
    try:
        response = openai.fine_tuning.jobs.create(
            training_file=training_file_id,
            model=model,
            suffix="Vaishnavi Model" # Adding the suffix directly in the fine-tuning job
        )
        print(f"Fine-tuning job started: {response.id}")
        return response.id
    except Exception as e:
        print(f"Error starting fine-tuning: {e}")
        return None

```


Step 4: Then, monitor this fine-tuning process. This may take few minutes to complete.



```
Fine_Tune.py > main
64
65 # Function to monitor the fine-tuning job and save metrics to a CSV file
66 def monitor_and_save(job_id, output_csv):
67     """
68     Monitors the fine-tuning job's status. Once the job is completed,
69     it saves the metrics (like loss and accuracy) to a CSV file.
70     """
71     try:
72         while True:
73             job_status = openai.fine_tuning.jobs.retrieve(job_id)
74             # Check the job's status
75             if job_status.status == 'succeeded':
76                 print("Fine-tuning completed successfully!")
77                 events = openai.fine_tuning.jobs.list_events(job_id)
78                 save_metrics_to_csv(events, output_csv) # Save metrics to CSV
79                 break
80             elif job_status.status == 'failed':
81                 print("Fine-tuning failed!")
82                 break
83             else:
84                 print(f"Fine-tuning in progress... (status: {job_status.status})")
85                 time.sleep(60) # Wait for 60 seconds before checking again
86     except Exception as e:
87         print(f"Error monitoring job: {e}")
88
```

Step 5: Once the fine-tuning process is successful, Save the metrics to a csv file.

```
Fine_Tune.py > main
89 # Function to save metrics to a CSV file
90 def save_metrics_to_csv(events, output_csv):
91     """
92     Saves the fine-tuning job's metrics (like training loss, sequence accuracy, token accuracy) to a CSV file.
93     """
94     try:
95         with open(output_csv, mode='w', newline='') as file:
96             writer = csv.writer(file)
97             # Write CSV header
98             writer.writerow(["step", "train_loss", "total_steps", "train_mean_token_accuracy"])
99
100         # Iterate through the events and extract metrics
101         for event in events:
102             if event.type == 'metrics': # Only process metric events
103                 metrics = event.data # Extract metric data
104                 writer.writerow([
105                     metrics.get('step', 'N/A'),
106                     metrics.get('train_loss', 'N/A'),
107                     metrics.get('total_steps', 'N/A'),
108                     metrics.get('train_mean_token_accuracy', 'N/A')
109                 ])
110         print(f"Metrics successfully saved to {output_csv}")
111     except Exception as e:
112         print(f"Error saving metrics to CSV: {e}")
113
```


Fine_tuning_metrics.csv

 fine_tuning_metrics.csv X

 fine_tuning_metrics.csv >  data

1	step,train_loss,total_steps,train_mean_token_accuracy
47	54,0.1215573251247406,99,0.9523809552192688
48	53,0.19870059192180634,99,0.9166666865348816
49	52,0.07533347606658936,99,0.9375
50	51,0.0041697025299072266,99,1.0
51	50,0.09862694144248962,99,1.0
52	49,0.2384372055530548,99,0.8823529481887817
53	48,0.17555296421051025,99,0.9473684430122375
54	47,0.16324013471603394,99,1.0
55	46,0.1669486165046692,99,1.0
56	45,0.35301464796066284,99,0.8999999761581421
57	44,0.31195369362831116,99,0.9523809552192688
58	43,0.42186903953552246,99,0.8823529481887817
59	42,0.059812165796756744,99,0.949999988079071
60	41,0.31486380100250244,99,0.8947368264198303
61	40,0.357208251953125,99,0.9230769276618958
62	39,0.3834072947502136,99,0.8461538553237915
63	38,0.26170656085014343,99,0.8461538553237915
64	37,0.03665351867675781,99,1.0
65	36,0.5882760882377625,99,0.8500000238418579
66	35,0.1965879201889038,99,1.0
67	34,0.4812479019165039,99,0.8333333134651184
68	33,0.7063129544258118,99,0.8500000238418579
69	32,0.5363132357597351,99,0.8947368264198303
70	31,0.6552855372428894,99,0.7916666865348816
71	30,0.6168771982192993,99,0.7692307829856873
72	29,0.6708090305328369,99,0.8823529481887817
73	28,0.12238538265228271,99,1.0
74	27,0.7210091948509216,99,0.7692307829856873
75	26,0.5629405975341797,99,0.8125
76	25,0.42375800013542175,99,0.7692307829856873
77	24,0.6360242962837219,99,0.8095238208770752

Output of the entire script:

```
(venv) vaishnavi@DESKTOP-9V8KJG2:/mnt/c/Users/Mohit/Desktop/Gen AI/Week 8$ python3 Fine_Tune.py
Conversion successful! data.json has been converted to data_prepared.jsonl
File uploaded: file-gUpBN7H9NmGE06KPipPD1WYP
Fine-tuning job started: ftjob-5rrKtbTTUUiHnoVNd2SNyEmq
Fine-tuning in progress... (status: validating_files)
Fine-tuning in progress... (status: running)
Fine-tuning in progress... (status: running)
Fine-tuning in progress... (status: running)
Fine-tuning in progress... (status: running)
Fine-tuning in progress... (status: running)
Fine-tuning in progress... (status: running)
Fine-tuning completed successfully!
Metrics successfully saved to fine_tuning_metrics.csv
(venv) vaishnavi@DESKTOP-9V8KJG2:/mnt/c/Users/Mohit/Desktop/Gen AI/Week 8$
```

GitHub Link:

<https://github.com/vaishnavi477/Machine-Learning/tree/main/Generative%20AI/Fine%20Tuning%20openai%20jsonl>

Google Slide Link:

https://docs.google.com/presentation/d/1ZFdLvJfd-IkDksu5gAr_1KOlrM6pdtTMVlwpEFFeDpA/edit?usp=sharing