Simon Tesfatsion

Fine-Tuning a Model for Drug and Malady Classification

GItHUB LInk

Google Slides Link

OVERVIEW

Introduction

What is Fine-Tuning?

- Fine-tuning customizes a base model (like GPT-3.5 Turbo) for specific tasks.
- It leverages a domain-specific dataset to improve accuracy and relevance.
- Enables the model to better understand niche vocabulary, patterns, and logic.

Dataset Preparation

Format: JSONL (JSON Lines).

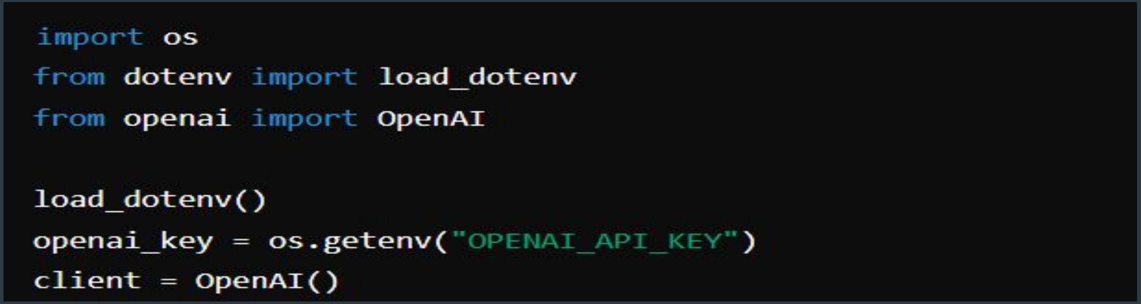
Contains drug names and corresponding maladies (classes).

Preprocessing steps:

1. Cleaning the data.
2. Ensuring consistency in labels.
3. Splitting the data for training and validation.

Setting Up Environment

- Tools and libraries used:
- os, dotenv: Environment variable management.
- openai: Access OpenAI API.
- Steps:
    - Install necessary libraries.
    - Load .env file containing API keys.
    - Initialize OpenAI client.

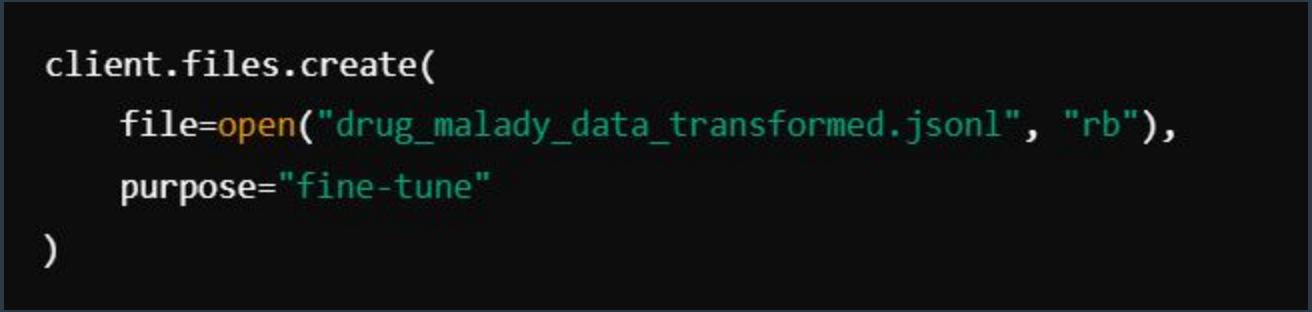


```python
import os
from dotenv import load_dotenv
from openai import OpenAI

load_dotenv()
openai_key = os.getenv("OPENAI_API_KEY")
client = OpenAI()
```

Uploading Dataset

Code to upload the dataset

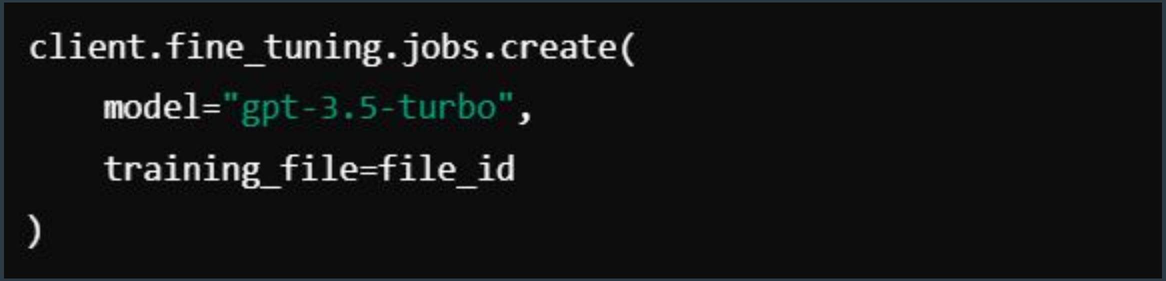


```
client.files.create(
    file=open("drug_malady_data_transformed.jsonl", "rb"),
    purpose="fine-tune"
)
```

Ensure the file is properly formatted and ready for use.

Fine-Tuning

- Create the fine-tuning job using

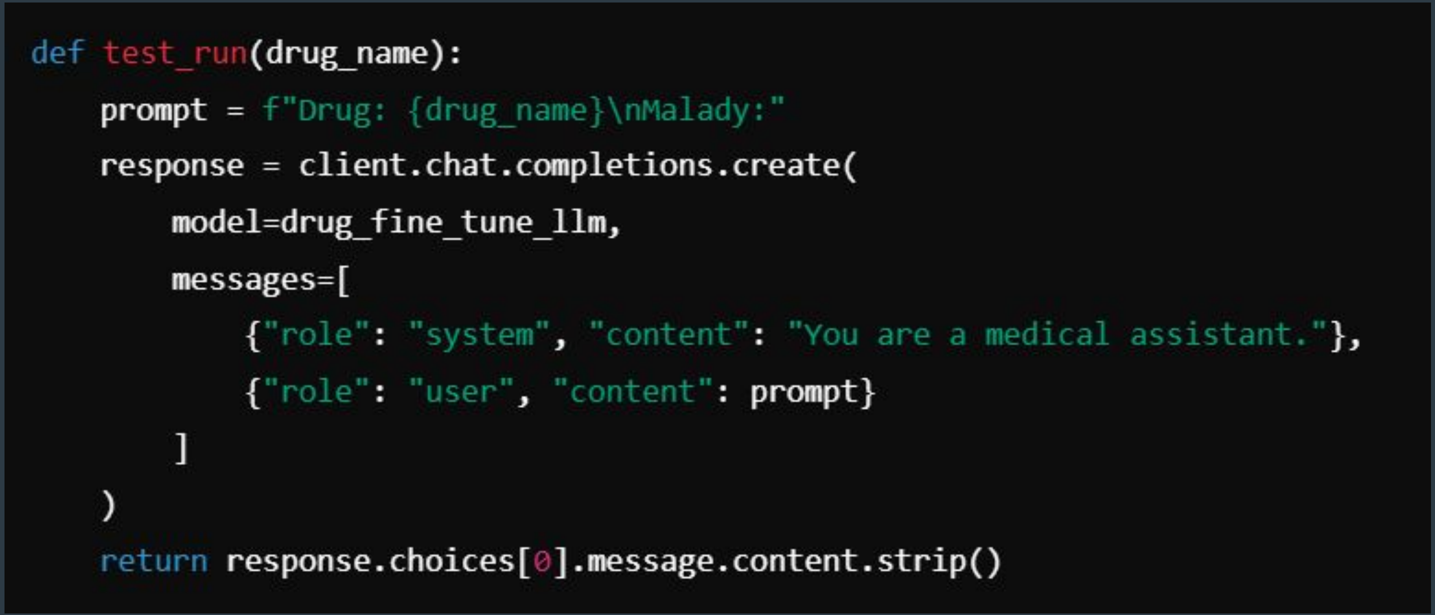


```
client.fine_tuning.jobs.create(
    model="gpt-3.5-turbo",
    training_file=file_id
)
```

- Track the job status to ensure completion.
- Fine-tuned model is saved under a unique ID

Testing the Fine-Tuned Model

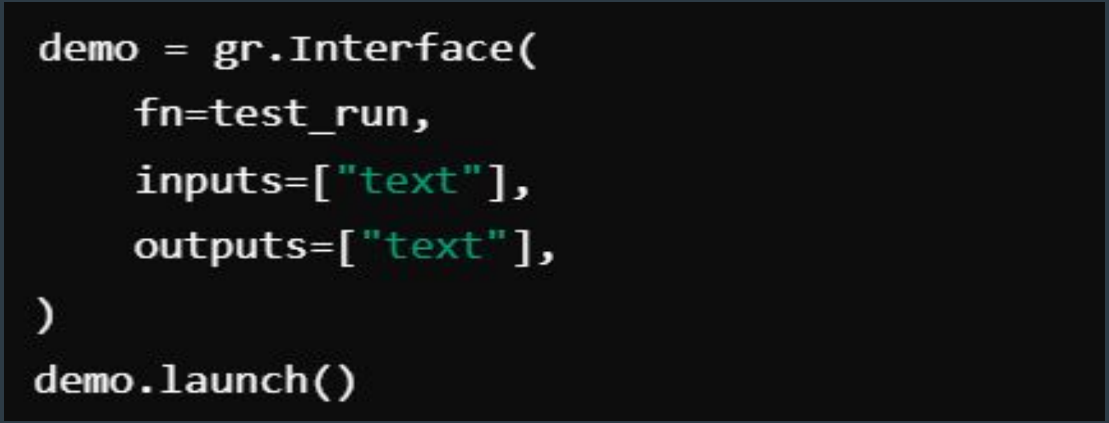
- Test the model's predictions by providing drug names.
- Use fine-tuned model ID to invoke predictions.
- Code for testing:



```python
def test_run(drug_name):
    prompt = f"Drug: {drug_name}\nMalady:"
    response = client.chat.completions.create(
        model=drug_fine_tune_llm,
        messages=[
            {"role": "system", "content": "You are a medical assistant."},
            {"role": "user", "content": prompt}
        ]
    )
    return response.choices[0].message.content.strip()
```

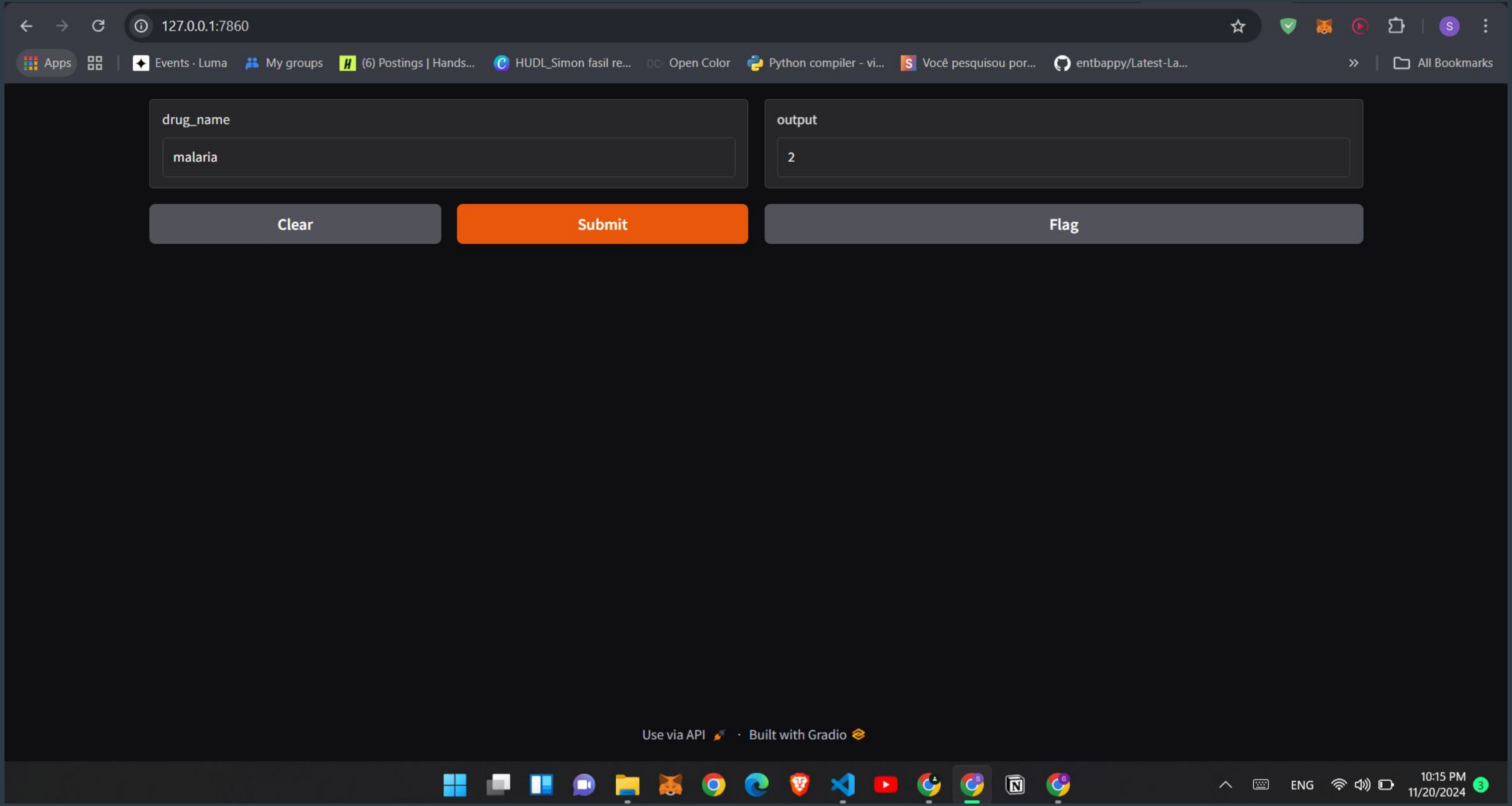
Building a User Interface

- Gradio simplifies model testing with a web-based UI.
- Input: Drug name.
- Output: Predicted drug class or malady.
- Code snippet:



```
demo = gr.Interface(
    fn=test_run,
    inputs=["text"],
    outputs=["text"],
)
demo.launch()
```

Live Demo:



drug_name

malaria

output

2

Clear

Submit

Flag

Use via API · Built with Gradio

Project Reference Materials

GitHub Link :

https://github.com/Montegan/drug_fintuned

Google Slides Link :

https://docs.google.com/presentation/d/14pGa3vzO6w2SvJkJ9FKEojE4KF2Zzp2m
xyxUUBobIJU/edit?usp=sharing

“

Thank You

Simon Tesfatsion