Exercises for Chapter 11: Advanced Fine Tuning: Drug Classification

Fine-Tuning based on 2000 drug examples from an Excel file

Step 1: Preparing the Data and Launching the Fine Tuning

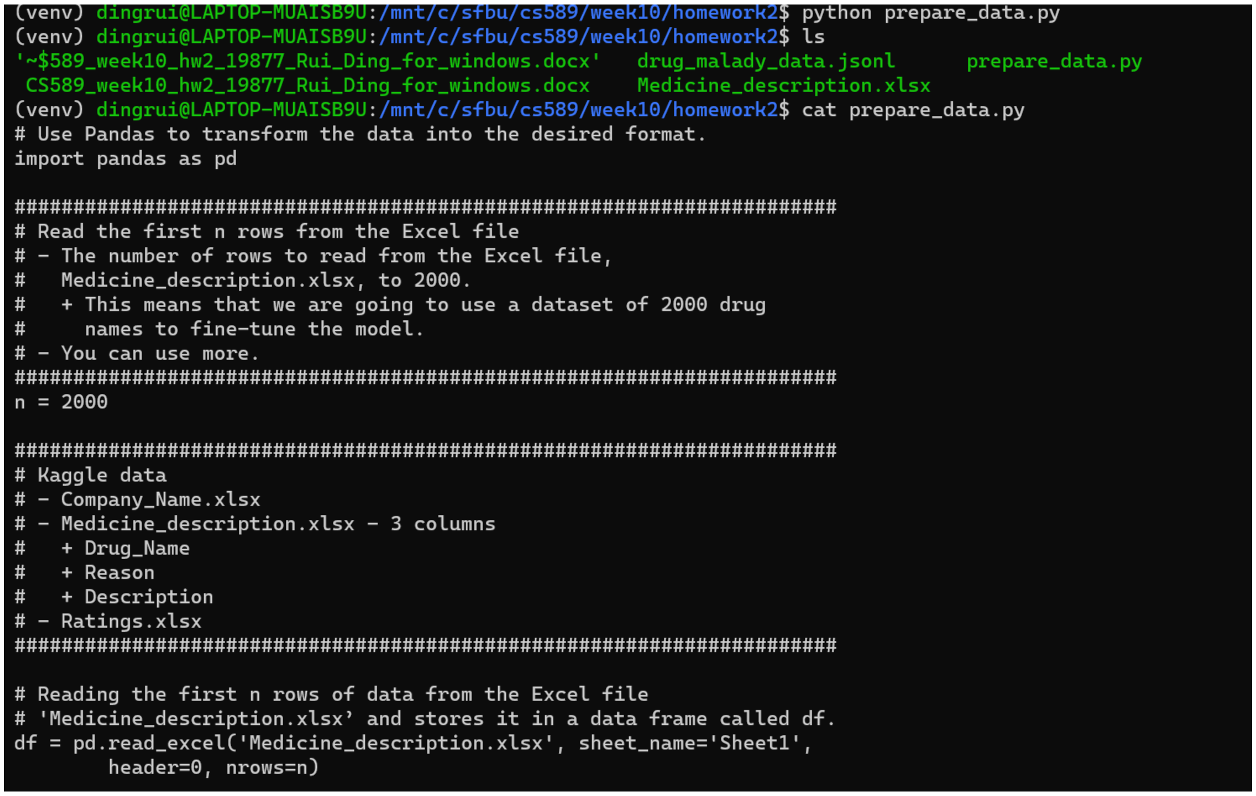
References

source

Step 1.1 Preparing the Data and Launching the Fine Tuning

```
# Use Pandas to transform the data into the desired format.
import pandas as pd

######################################################################
# Read the first n rows from the Excel file
# - The number of rows to read from the Excel file,
#   Medicine_description.xlsx, to 2000.
#   + This means that we are going to use a dataset of 2000 drug
#     names to fine-tune the model.
# - You can use more.
######################################################################
n = 2000

######################################################################
# Kaggle data
# - Company Name.xlsx
# - Medicine_description.xlsx - 3 columns
#   + Drug_Name
#   + Reason
#   + Description
# - Ratings.xlsx
######################################################################

# Reading the first n rows of data from the Excel file
# 'Medicine_description.xlsx' and stores it in a data frame called df.
df = pd.read_excel('Medicine_description.xlsx', sheet_name='Sheet1',
        header=0, nrows=n)

# Get the unique values in the 'Reason' column of the data frame,
# stores them in an array called reasons
reasons = df["Reason"].unique()
```

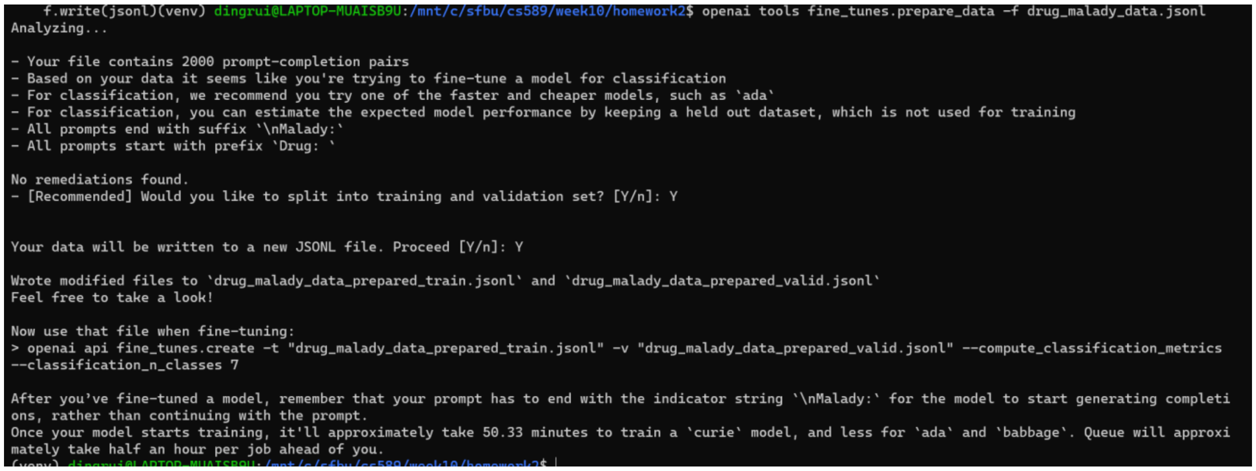
```
# Assigns a numerical index to each unique value in the reasons
# array, and stores it in a dictionary called reasons_dict.
reasons_dict = {reason: i for i, reason in enumerate(reasons)}

# Add a new line and “Malady:” to the end of each drug name in
# the ‘Drug_Name’ column of the data frame.
# - The desired format:
#       Drug: <Drug_Name>\nMalady:
df["Drug_Name"] = "Drug: " + df["Drug_Name"] + "\n" + "Malady:"

# It concatenates a space and the corresponding numerical index
# from the reasons_dict to the end of each 'Reason’
# value in the data frame.
df["Reason"] = " " + df["Reason"].apply(lambda x: "" + str(reasons_dict[x]))

# For this example, we don’t need the ‘Description’ column, that’s
# why the script drops it from the data frame.
df.drop(["Description"], axis=1, inplace=True)

# Renaming the ‘Drug_Name’ column to ‘prompt’
# and the ‘Reason’ column to ‘completion’.
df.rename(columns={"Drug_Name": "prompt", "Reason": "completion"},
inplace=True)

# Convert the dataframe to jsonl format
jsonl = df.to_json(orient="records", indent=0, lines=True)

# Write the jsonl to a file
#
# - drug_malady_data.jsonl has data like
#    [..]
#    {"prompt":"Drug: Acleen 1% Lotion 25ml\nMalady:","completion":" 0"}
#    [..]
#    {"prompt":"Drug: Capnea Injection 1ml\nMalady:","completion":" 1"}
#    [..]
#    {"prompt":"Drug: Mondeslor Tablet 10'S\nMalady:","completion":" 2"}
#    [..]
with open("drug_malady_data.jsonl", "w") as f:
    f.write(jsonl)
```



```
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ python prepare_data.py
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ ls
'~$589_week10_hw2_19877_Rui_Ding_for_windows.docx'   drug_malady_data.jsonl      prepare_data.py
 CS589_week10_hw2_19877_Rui_Ding_for_windows.docx    Medicine_description.xlsx
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ cat prepare_data.py
# Use Pandas to transform the data into the desired format.
import pandas as pd

######################################################################
# Read the first n rows from the Excel file
# - The number of rows to read from the Excel file,
#   Medicine_description.xlsx, to 2000.
#   + This means that we are going to use a dataset of 2000 drug
#     names to fine-tune the model.
# - You can use more.
######################################################################
n = 2000

######################################################################
# Kaggle data
# - Company_Name.xlsx
# - Medicine_description.xlsx - 3 columns
#   + Drug_Name
#   + Reason
#   + Description
# - Ratings.xlsx
######################################################################

# Reading the first n rows of data from the Excel file
# 'Medicine_description.xlsx' and stores it in a data frame called df.
df = pd.read_excel('Medicine_description.xlsx', sheet_name='Sheet1',
        header=0, nrows=n)
```

Step1.2 Command to Prepare Data

Analyze and prepare the data using the OpenAI tools fine_tunes.prepare_data command.



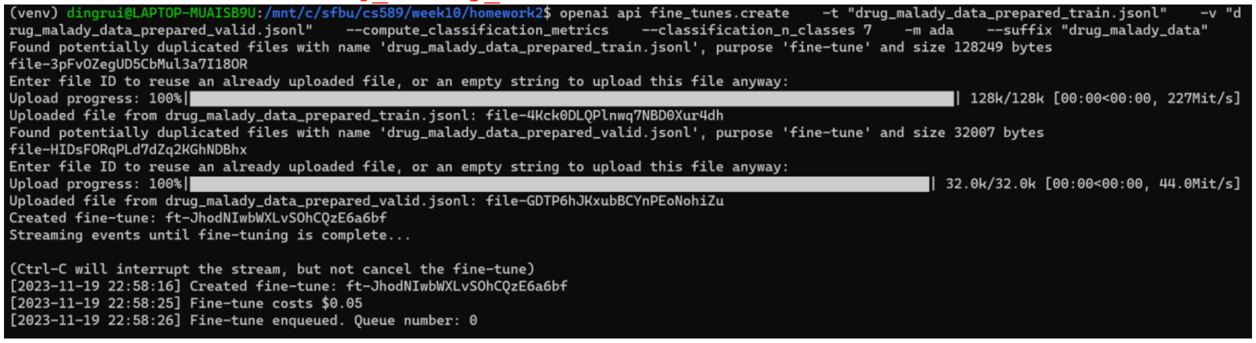
```
    f.write(jsonl)(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ openai tools fine_tunes.prepare_data -f drug_malady_data.jsonl
Analyzing...

- Your file contains 2000 prompt-completion pairs
- Based on your data it seems like you're trying to fine-tune a model for classification
- For classification, we recommend you try one of the faster and cheaper models, such as `ada`
- For classification, you can estimate the expected model performance by keeping a held out dataset, which is not used for training
- All prompts end with suffix `\nMalady:`
- All prompts start with prefix `Drug: `

No remediations found.
- [Recommended] Would you like to split into training and validation set? [Y/n]: Y


Your data will be written to a new JSONL file. Proceed [Y/n]: Y

Wrote modified files to `drug_malady_data_prepared_train.jsonl` and `drug_malady_data_prepared_valid.jsonl`
Feel free to take a look!

Now use that file when fine-tuning:
> openai api fine_tunes.create -t "drug_malady_data_prepared_train.jsonl" -v "drug_malady_data_prepared_valid.jsonl" --compute_classification_metrics
--classification_n_classes 7

After you’ve fine-tuned a model, remember that your prompt has to end with the indicator string `\nMalady:` for the model to start generating completi
ons, rather than continuing with the prompt.
Once your model starts training, it'll approximately take 50.33 minutes to train a `curie` model, and less for `ada` and `babbage`. Queue will approxi
mately take half an hour per job ahead of you.
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$
```

Step 1.3 Command to train the model

Use the provided command to train the model using fine_tunes.create.

```
# Export your OpenAI key
export OPENAI_API_KEY="xxxxxxxxxxxx"
```

```
openai api fine_tunes.create \
   -t "drug_malady_data_prepared_train.jsonl" \
   -v "drug_malady_data_prepared_valid.jsonl" \
   --compute_classification_metrics \
   --classification_n_classes 3 \
   -m ada \
   --suffix "drug_malady_data"
```



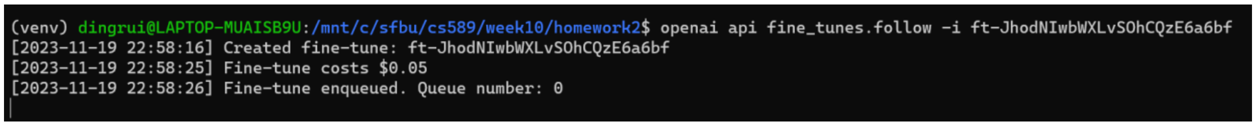
```
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ openai api fine_tunes.create    -t "drug_malady_data_prepared_train.jsonl"    -v "d
rug_malady_data_prepared_valid.jsonl"    --compute_classification_metrics    --classification_n_classes 7    -m ada    --suffix "drug_malady_data"
Found potentially duplicated files with name 'drug_malady_data_prepared_train.jsonl', purpose 'fine-tune' and size 128249 bytes
file-3pFvOZegUD5CbMul3a7I18OR
Enter file ID to reuse an already uploaded file, or an empty string to upload this file anyway:
Upload progress: 100%|████████████████████████████████████████| 128k/128k [00:00<00:00, 227Mit/s]
Uploaded file from drug_malady_data_prepared_train.jsonl: file-4Kck0DLQPlnwq7NBD0Xur4dh
Found potentially duplicated files with name 'drug_malady_data_prepared_valid.jsonl', purpose 'fine-tune' and size 32007 bytes
file-HIDsFORqPLd7dZq2KGhNDBhx
Enter file ID to reuse an already uploaded file, or an empty string to upload this file anyway:
Upload progress: 100%|██████████████████████████████████████| 32.0k/32.0k [00:00<00:00, 44.0Mit/s]
Uploaded file from drug_malady_data_prepared_valid.jsonl: file-GDTP6hJKxubBCYnPEoNohiZu
Created fine-tune: ft-JhodNIwbWXLvSOhCQzE6a6bf
Streaming events until fine-tuning is complete...

(Ctrl-C will interrupt the stream, but not cancel the fine-tune)
[2023-11-19 22:58:16] Created fine-tune: ft-JhodNIwbWXLvSOhCQzE6a6bf
[2023-11-19 22:58:25] Fine-tune costs $0.05
[2023-11-19 22:58:26] Fine-tune enqueued. Queue number: 0
```

Step 1.4 Checking Job Progress

If the client disconnects during fine-tuning, use the following command to check job progress.

```
openai api fine_tunes.follow -i <JOB ID>
```

The output will display progress information and queue numbers.

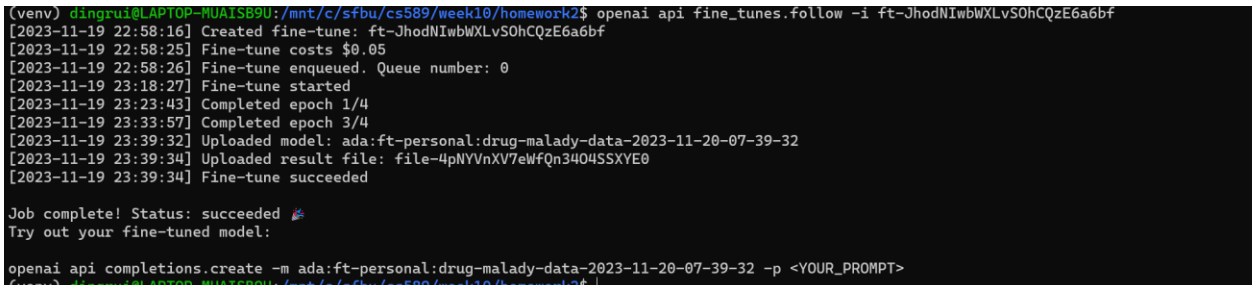


```
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ openai api fine_tunes.follow -i ft-JhodNIwbWXLvSOhCQzE6a6bf
[2023-11-19 22:58:16] Created fine-tune: ft-JhodNIwbWXLvSOhCQzE6a6bf
[2023-11-19 22:58:25] Fine-tune costs $0.05
[2023-11-19 22:58:26] Fine-tune enqueued. Queue number: 0
```

Step 1.5 Completion of Fine-Tuning

When the fine-tuning job is completed, you'll receive an output like this:

```
Created fine-tune: <JOB ID>
Fine-tune costs $0.03
Fine-tune enqueued

Fine-tune is in the queue. Queue number: 31
Fine-tune is in the queue. Queue number: 30
Fine-tune is in the queue. Queue number: 29
Fine-tune is in the queue. Queue number: 28
[...]
[...]
[...]
Fine-tune is in the queue. Queue number: 2
Fine-tune is in the queue. Queue number: 1
Fine-tune is in the queue. Queue number: 0
Fine-tune started
Completed epoch 1/4
Completed epoch 2/4
Completed epoch 3/4
Completed epoch 4/4
Uploaded model: <MODEL ID>
Uploaded result file: <FILE ID>
Fine-tune succeeded

Job complete! Status: succeeded

Try out your fine-tuned model:

openai api completions.create -m <MODEL ID> -p <YOUR_PROMPT>
```



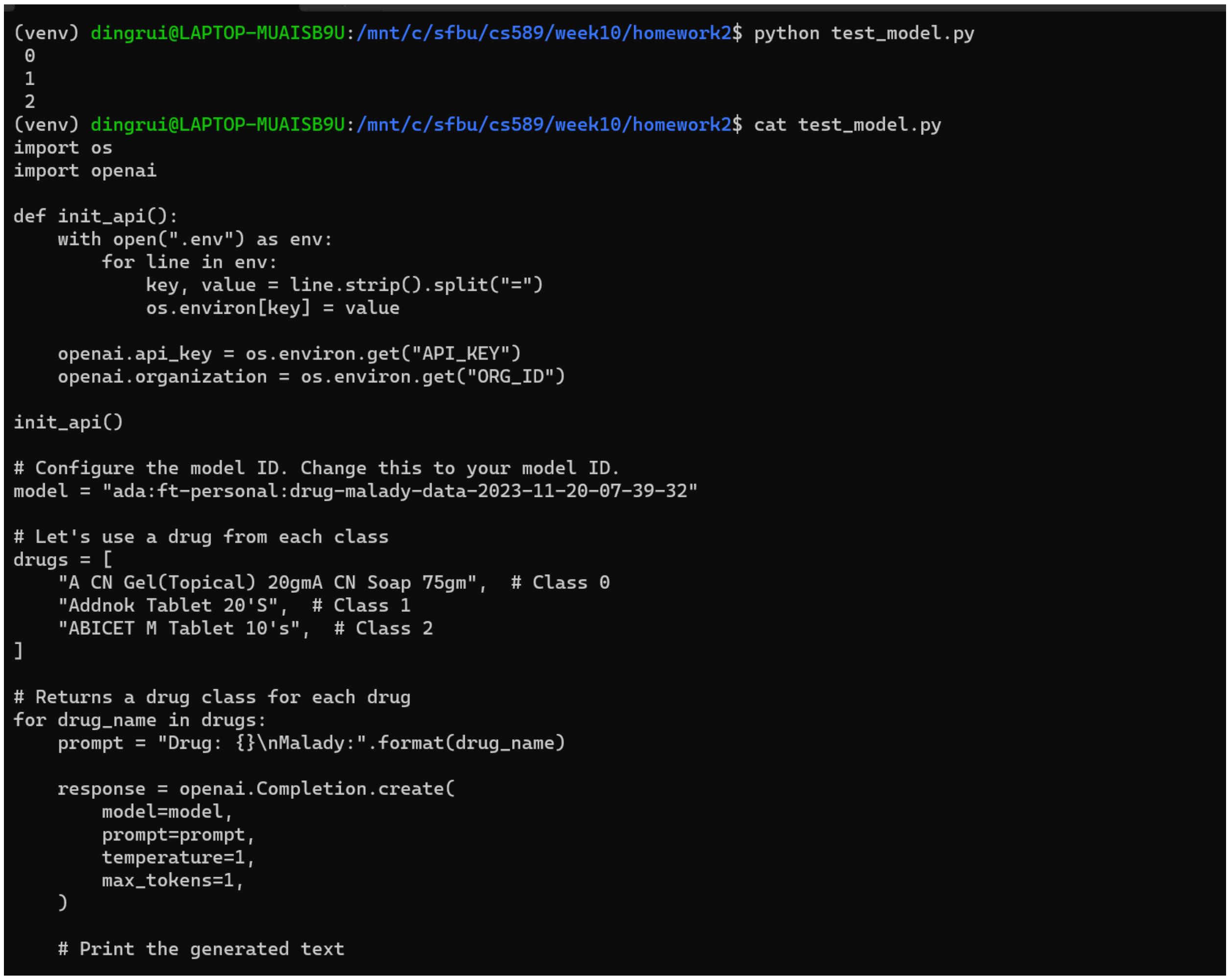
```
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ openai api fine_tunes.follow -i ft-JhodNIwbWXLvSOhCQzE6a6bf
[2023-11-19 22:58:16] Created fine-tune: ft-JhodNIwbWXLvSOhCQzE6a6bf
[2023-11-19 22:58:25] Fine-tune costs $0.05
[2023-11-19 22:58:26] Fine-tune enqueued. Queue number: 0
[2023-11-19 23:18:27] Fine-tune started
[2023-11-19 23:23:43] Completed epoch 1/4
[2023-11-19 23:33:57] Completed epoch 3/4
[2023-11-19 23:39:32] Uploaded model: ada:ft-personal:drug-malady-data-2023-11-20-07-39-32
[2023-11-19 23:39:34] Uploaded result file: file-4pNYVnXV7eWfQn34O4SSXYE0
[2023-11-19 23:39:34] Fine-tune succeeded

Job complete! Status: succeeded 🎉
Try out your fine-tuned model:

openai api completions.create -m ada:ft-personal:drug-malady-data-2023-11-20-07-39-32 -p <YOUR_PROMPT>
```

Step 2: Testing the Fine Tuned Model

References

source

Step 2.1 Python Code: Model Testing



```
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ python test_model.py
 0
 1
 2
(venv) dingrui@LAPTOP-MUAISB9U:/mnt/c/sfbu/cs589/week10/homework2$ cat test_model.py
import os
import openai

def init_api():
    with open(".env") as env:
        for line in env:
            key, value = line.strip().split("=")
            os.environ[key] = value

    openai.api_key = os.environ.get("API_KEY")
    openai.organization = os.environ.get("ORG_ID")

init_api()

# Configure the model ID. Change this to your model ID.
model = "ada:ft-personal:drug-malady-data-2023-11-20-07-39-32"

# Let's use a drug from each class
drugs = [
    "A CN Gel(Topical) 20gmA CN Soap 75gm",  # Class 0
    "Addnok Tablet 20'S",  # Class 1
    "ABICET M Tablet 10's",  # Class 2
]

# Returns a drug class for each drug
for drug_name in drugs:
    prompt = "Drug: {}\nMalady:".format(drug_name)

    response = openai.Completion.create(
        model=model,
        prompt=prompt,
        temperature=1,
        max_tokens=1,
    )

    # Print the generated text
```