ChatGPT_API

March 8, 2024

#Using the ChatGPT API: An example in drug synergy prediction

In this notebook, we'll explore how to interact with OpenAI's ChatGPT API.

Before we start, you need to have:

- An OpenAI account.
- API key from OpenAI.

First, install and import the necessary packages:

Section 1: Install OpenAI and Load Dataset

[ ]:
```
!pip install openai
```

```
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages
(0.28.1)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-
packages (from openai) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from openai) (4.66.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from openai) (3.8.6)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.3.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.20->openai) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai)
(2023.7.22)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->openai) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (4.0.3)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->openai) (1.9.2)
```

```
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.3.1)
```

[ ]:
```
import pandas as pd
import numpy as np
import json
import time
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, average_precision_score,␣
 ↪RocCurveDisplay
from torch.utils.data import Dataset, DataLoader
import openai

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

Initialize the OpenAI Client with your API Key

[ ]:
```
openai.api_key = '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' # For␣
 ↪demo only. Replace it with your own API.
```

Load the Drug Synergy Dataset

[ ]:
```
df = pd.read_csv('drive/MyDrive/AIHealthTutorial/Online/LLM/data_synergy.csv')
```

[ ]:
```
df.iloc[:,[0,2,4,5,6,7,8]]
```

[ ]:

	drug_row	drug_col	cell_line_name	tissue_name	ri_row	ri_col	synergy_loewe
0	lonidamine	717906-29-1	A-673	bone	0.568	28.871	-11.702283
1	Ethyl bromopyruvate	717906-29-1	A-673	bone			
2	Tranilast (trans-)	717906-29-1	A-673	bone			
3	Lenalidomide	717906-29-1	A-673	bone			
4	Pomalidomide	717906-29-1	A-673	bone			
…	…	…	…	…			
717997	AZD4547	PF-04691502	SW780FGF	urinary_tract			
717998	AZD4547	AZD1208	SW780FGF	urinary_tract			
717999	AZD4547	Carfilzomib	SW780FGF	urinary_tract			
718000	AZD4547	Cediranib	SW780FGF	urinary_tract			
718001	AZD5582	TNF-Alpha	SW780FGF	urinary_tract			

```
1         4.282  26.716     -16.185120
2         3.056  24.391     -16.588246
3        -4.751  23.131     -10.877569
4         2.972  19.578      -1.901326
…           …     …              …
717997   19.451  19.455      -7.375586
717998    2.659  -1.894      -3.936333
717999   19.774  56.542      -1.732535
718000   16.433  14.960       1.867163
718001   81.480  91.043       3.923324

[718002 rows x 7 columns]
```

[ ]:
```
df
```

[ ]:
```
                   drug_row  \
0                lonidamine
1        Ethyl bromopyruvate
2         Tranilast (trans-)
3               Lenalidomide
4               Pomalidomide
…                        …
717997               AZD4547
717998               AZD4547
717999               AZD4547
718000               AZD4547
718001               AZD5582

                                                 smiles_row     drug_col  \
0       C1=CC=C2C(=C1)C(=NN2CC3=C(C=C(C=C3)Cl)Cl)C(=O)O\n  717906-29-1
1                                  CCOC(=O)C(=O)CBr\n  717906-29-1
2         COC1=C(C=C(C=C1)C=CC(=O)NC2=CC=CC=C2C(=O)O)OC\n  717906-29-1
3                 C1CC(=O)NC(=O)C1N2CC3=C(C2=O)C=CC=C3N\n  717906-29-1
4           C1CC(=O)NC(=O)C1N2C(=O)C3=C(C2=O)C(=CC=C3)N\n  717906-29-1
…                                                     …           …
717997  CC1CN(CC(N1)C)C2=CC=C(C=C2)C(=O)NC3=NNC(=C3)CC…  PF-04691502
717998  CC1CN(CC(N1)C)C2=CC=C(C=C2)C(=O)NC3=NNC(=C3)CC…      AZD1208
717999  CC1CN(CC(N1)C)C2=CC=C(C=C2)C(=O)NC3=NNC(=C3)CC…  Carfilzomib
718000  CC1CN(CC(N1)C)C2=CC=C(C=C2)C(=O)NC3=NNC(=C3)CC…    Cediranib
718001  CC(C(=O)NC(C1CCCCC1)C(=O)N2CCCC2C(=O)NC3C(CC4=…    TNF-Alpha

                                                 smiles_col cell_line_name  \
0       CN(C1=CC=CC=C1CNC2=NC(=NC=C2C(F)(F)F)NC3=CC4=C…          A-673
1       CN(C1=CC=CC=C1CNC2=NC(=NC=C2C(F)(F)F)NC3=CC4=C…          A-673
2       CN(C1=CC=CC=C1CNC2=NC(=NC=C2C(F)(F)F)NC3=CC4=C…          A-673
3       CN(C1=CC=CC=C1CNC2=NC(=NC=C2C(F)(F)F)NC3=CC4=C…          A-673
4       CN(C1=CC=CC=C1CNC2=NC(=NC=C2C(F)(F)F)NC3=CC4=C…          A-673
```

```
…                                                       …            …
717997  CC1=C2C=C(C(=O)N(C2=NC(=N1)N)C3CCC(CC3)OCCO)C4…       SW780FGF
717998  C1CC(CN(C1)C2=C(C=CC=C2C3=CC=CC=C3)C=C4C(=O)NC…       SW780FGF
717999  CC(C)CC(C(=O)C1(CO1)C)NC(=O)C(CC2=CC=CC=C2)NC(…       SW780FGF
718000  CC1=CC2=C(N1)C=CC(=C2F)OC3=NC=NC4=CC(=C(C=C43)…       SW780FGF
718001  CC(C)CC(C(=O)NC(CO)C(=O)O)NC(=O)C(CCCN=C(N)N)N…       SW780FGF

          tissue_name  ri_row  ri_col  synergy_loewe
0               bone   0.568  28.871     -11.702283
1               bone   4.282  26.716     -16.185120
2               bone   3.056  24.391     -16.588246
3               bone  -4.751  23.131     -10.877569
4               bone   2.972  19.578      -1.901326
…                …      …      …              …
717997  urinary_tract  19.451  19.455      -7.375586
717998  urinary_tract   2.659  -1.894      -3.936333
717999  urinary_tract  19.774  56.542      -1.732535
718000  urinary_tract  16.433  14.960       1.867163
718001  urinary_tract  81.480  91.043       3.923324

[718002 rows x 9 columns]
```

Section 2: Zero-shot ChatGPT Prompt Engineering

0.1 Making a Simple Request, just like ChatGPT interface

0.1.1 One-time Request and Responce

Prompt: Decide in a single word if the synergy of the drug combination in the cell line is positive (synergy >=5) or negative (synergy <5). Drug combination and cell line: The first drug is AZD4877. The second drug is AZD1208. The cell line is T24. Tissue is bone. The first drug's sensitivity using relative inhibition is 99.091. The second drug's sensitivity using relative inhibition is 3.803. Is this drug combination synergy positive or negative?

[ ]:
```
response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[
        {"role": "user", "content": "Decide in a single word if the synergy of␣
 ↪the drug combination in the cell line is positive (synergy >=5) or negative␣
 ↪(synergy <5). Drug combination and cell line: The first drug is AZD4877. The␣
 ↪second drug is AZD1208. The cell line is T24. Tissue is bone. The first␣
 ↪drug's sensitivity using relative inhibition is 99.091. The second drug's␣
 ↪sensitivity using relative inhibition is 3.803. Is this drug combination␣
 ↪synergy positive or negative?"},
    ]
)

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

```
Positive
```

Get the ground truth synergy from the dataset

[ ]:
```
df.
  ↪loc[(df['drug_row']=='AZD-4877')&(df['drug_col']=='AZD1208')&(df['cell_line_name']=='T24')]
```

Using the Chat-based Approach

Role: You are an expert on drug discovery.

Prompt: Decide in a single word if the synergy of the drug combination in the cell line is positive (synergy >=5) or negative (synergy <5). Drug combination and cell line: The first drug is AZD4877. The second drug is AZD1208. The cell line is T24. Tissue is bone. The first drug's sensitivity using relative inhibition is 99.091. The second drug's sensitivity using relative inhibition is 3.803. Is this drug combination synergy positive or negative?

[ ]:
```
response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[
        {"role": "system", "content": "You are an expert on drug discovery."},
        {"role": "user", "content": "Decide in a single word if the synergy of␣
 ↪the drug combination in the cell line is positive (synergy >=5) or negative␣
 ↪(synergy <5). Drug combination and cell line: The first drug is AZD4877. The␣
 ↪second drug is AZD1208. The cell line is T24. Tissue is bone. The first␣
 ↪drug's sensitivity using relative inhibition is 99.091. The second drug's␣
 ↪sensitivity using relative inhibition is 3.803. Is this drug combination␣
 ↪synergy positive or negative?"},
    ]
)

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

```
Positive
```

0.1.2 Continuing a Conversation

New prompt: Can you provide details why the two drugs are synergistic in the cell line?

[ ]:
```
messages = [
    {"role": "system", "content": "You are an expert on drug discovery."},
    {"role": "user", "content": "Decide in a single word if the synergy of the␣
 ↪drug combination in the cell line is positive (synergy >=5) or negative␣
 ↪(synergy <5). Drug combination and cell line: The first drug is AZD4877. The␣
 ↪second drug is AZD1208. The cell line is T24. Tissue is bone. The first␣
 ↪drug's sensitivity using relative inhibition is 99.091. The second drug's␣
 ↪sensitivity using relative inhibition is 3.803. Is this drug combination␣
 ↪synergy positive or negative?"},
    {"role": "assistant", "content": response.choices[0].message['content']},
```

```
    {"role": "user", "content": "Can you provide details why the two drugs are␣
  ↪synergistic in the cell line?"},
]

response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=messages
)

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

```
Yes. In drug synergy, the combined effectiveness of two drugs is determined not
merely by their individual efficacy but also by how they interact with each
other when used together. Here, the first drug, AZD4877 shows a high sensitivity
in the T24 cell line (from a bone tissue) as indicated by the high relative
inhibition value of 99.091. This means that it is highly effective in preventing
the growth of the cells. On the other hand, the second drug, AZD1208, shows a
lower relative inhibition value of 3.803, indicating that its effectiveness is
lower comparably.

However, in drug synergy, even a low efficacy drug can contribute to significant
improvements when combined with a high efficacy drug. It's the combined effect
that creates the synergy. Therefore, even though AZD1208 shows lower
sensitivity, its combination with AZD4877 could increase its overall efficacy,
making the combined effect more significant than their individual effects. This
explains the positive synergy between these two drugs in the T24 cell line.
```

Section 3: Getting prompts for training and test data for endometrium

Split train and test set, use “endometrium” as an example

[ ]:
```
data_endometrium = df.loc[df['tissue_name']=='endometrium']
data_index = list(data_endometrium.index)
train_index, test_index = train_test_split(data_index, test_size=0.2,␣
  ↪random_state=42)
```

[ ]:
```
print(train_index)
print(test_index)
```

```
[100214, 100215, 100209, 100167, 100202, 100191, 100196, 100179, 100205, 100193,
100208, 100190, 100173, 100223, 100200, 100163, 100177, 100194, 100168, 100204,
100166, 100216, 100226, 100175, 100187, 100186, 100184, 100227, 100171, 100192,
100224, 100210, 100197, 100189, 100203, 100213, 100161, 100181, 100162, 100222,
100199, 100195, 100212, 100183, 100219, 100170, 100182, 100178, 100217, 100198,
100180, 100220, 100174, 100211]
[100206, 100176, 100164, 100169, 100188, 100201, 100218, 100165, 100221, 100172,
100185, 100225, 100207, 100160]
```

To make it consistent, we use a pre-defined split train and test sets about endometrium (80% for

training, the file is shared in the folder)

[ ]:
```
with open('drive/MyDrive/AIHealthTutorial/Online/LLM/train_test_split.json',␣
↪'r') as f:
    data_split = json.load(f)
print(data_split['endometrium']['train'])
print(data_split['endometrium']['test'])
```

```
[100160, 100161, 100162, 100163, 100164, 100165, 100167, 100168, 100169, 100170,
100171, 100174, 100175, 100176, 100177, 100178, 100180, 100181, 100182, 100183,
100185, 100187, 100188, 100189, 100190, 100191, 100192, 100193, 100196, 100197,
100198, 100200, 100202, 100203, 100204, 100205, 100206, 100207, 100208, 100209,
100211, 100212, 100213, 100214, 100216, 100217, 100218, 100219, 100220, 100221,
100223, 100224, 100225, 100227]
[100166, 100172, 100173, 100179, 100184, 100186, 100194, 100195, 100199, 100201,
100210, 100215, 100222, 100226]
```

Write a function to get the prompt for each input

[ ]:
```
class DrugCombDataset(Dataset):
    def __init__(self, df):
        self.df = df

    def __len__(self):
        return len(self.df)

    def __getitem__(self, index):
        column_names = [
            ("drug_row", "The first drug is "),
            ("drug_col", ". The second drug is "),
            ("cell_line_name", ". Cell line is "),
            ("tissue_name", ". Tissue is "),
            ("ri_row", ". First drug's sensitivity score using relative␣
↪inhibition is "),
            ("ri_col", ". Second drug's sensitivity score using relative␣
↪inhibition is ")
        ]

        x_strs = [f"{col_desc}{self.df.iloc[index][col]}" for col, col_desc in␣
↪column_names]
        x_str = ''.join(x_strs)
        x_str = x_str.replace('\n', '')
        x_str = 'Decide in a single word if the synergy of the drug combination␣
↪in the cell line is positive or not. Synergy score more than or equal 5␣
↪means positive and synergy score less than 5 means negative. '+x_str
        x_str = x_str+'. Please decide whether the synergy is positive or␣
↪negative.'
```

```
        return x_str
```

Get the prompt for test dataset, using "endometrium" as an example

[ ]:
```
df['ri_row'] = df['ri_row'].astype(int)
df['ri_col'] = df['ri_col'].astype(int)
df['synergy_class'] = df['synergy_loewe'].apply(lambda x: x > 5).astype(int)

test_indices = data_split['endometrium']['test']
test_df = df.iloc[test_indices]
test_ds = DrugCombDataset(test_df)
```

Using GPT as a generative model to get the positive or negative results of synergy for testing data of endometrium

[ ]:
```
results = []
for prompt in tqdm(test_ds):
  response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
          {"role": "user", "content": prompt},
      ]
  )
  results.append(response.choices[0].message['content'])
  time.sleep(3)
```

```
100%|      | 14/14 [01:02<00:00,  4.48s/it]
```

[ ]:
```
results
```

[ ]:
```
['Positive',
 'positive',
 'negative',
 'positive',
 'negative',
 'negative',
 'negative',
 'Negative.',
 'Negative',
 'Positive',
 'positive',
 'positive',
 'positive',
 'Positive']
```

[ ]:
```
#manually change the positive result to 1 and negative result to 0. In this␣
 ↪case [1,1,0,1,0,0,0,0,0,1,1,1,1,1]
test_labels = list(df.iloc[test_indices]['synergy_class'])
```

```
test_pred = [1,1,0,1,0,0,0,0,0,1,1,1,1,1]
auroc = roc_auc_score(test_labels, test_pred)
auprc = average_precision_score(test_labels, test_pred)
print('\nAUROC:', auroc, '\nAUPRC', auprc)
```

```
AUROC: 0.6428571428571428
AUPRC 0.5892857142857143
```

Section 4: Using ChatGPT embeddings for synergy prediction

Write a function to get the embeddings for a dataset

[ ]:
```
def generate_embeddings(texts, model="text-embedding-ada-002"):
    embeddings = []
    for text in tqdm(texts):
        text = text.replace("\n", " ")
        response = openai.Embedding.create(input = [text], model=model)['data']
        embeddings.append(response[0]['embedding'])
    return np.array(embeddings)
```

Get the embeddings of prompts of training dataset of endometrium

[ ]:
```
train_indices = data_split['endometrium']['train']
train_df = df.iloc[train_indices]
train_ds = DrugCombDataset(train_df)

embeddings = generate_embeddings(train_ds)
```

```
100%|      | 54/54 [00:07<00:00,  7.25it/s]
```

Show the shape of the embeddings

[ ]:
```
np.shape(embeddings)
```

[ ]:
```
(54, 1536)
```

Get the synergy score (positve 1 or negative 0) from the training datasets

[ ]:
```
labels = list(df.iloc[train_indices]['synergy_class'])
```

Train a simple classifier to predict the positive or negative synergy using embeddings

[ ]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(embeddings, labels)
```

[ ]:
```
LogisticRegression(max_iter=1000)
```

Test the performance

```
[ ]: test_embeddings = generate_embeddings(test_ds)
test_labels = list(df.iloc[test_indices]['synergy_class'])

test_pred = model.predict_proba(test_embeddings)[:,1]
auroc = roc_auc_score(test_labels, test_pred)
auprc = average_precision_score(test_labels, test_pred)
print('\nAUROC:', auroc, '\nAUPRC', auprc)
```

```
100%|      | 14/14 [00:02<00:00,  6.42it/s]


AUROC: 0.8979591836734695
AUPRC 0.8736394557823128
```

Section 5: Using CancerGPT embedding for synergy prediction

CancerGPT is GPT2 (a much smaller model) finetuned on common cancers. It has the built in one layer MLP for classification, which will generate output as 1 (positive synergy) or 0 (negative synergy).

```
[ ]: !pip install transformers[torch]
```

```
Requirement already satisfied: transformers[torch] in
/usr/local/lib/python3.10/dist-packages (4.35.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers[torch]) (3.12.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in
/usr/local/lib/python3.10/dist-packages (from transformers[torch]) (0.17.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers[torch]) (1.23.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers[torch]) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers[torch]) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers[torch]) (2.31.0)
Requirement already satisfied: tokenizers<0.15,>=0.14 in
/usr/local/lib/python3.10/dist-packages (from transformers[torch]) (0.14.1)
Requirement already satisfied: safetensors>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from transformers[torch]) (0.4.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers[torch]) (4.66.1)
Requirement already satisfied: torch!=1.12.0,>=1.10 in
/usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2.1.0+cu118)
```

```
Requirement already satisfied: accelerate>=0.20.3 in
/usr/local/lib/python3.10/dist-packages (from transformers[torch]) (0.24.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages
(from accelerate>=0.20.3->transformers[torch]) (5.9.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from huggingface-hub<1.0,>=0.16.4->transformers[torch]) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.16.4->transformers[torch]) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch!=1.12.0,>=1.10->transformers[torch]) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch!=1.12.0,>=1.10->transformers[torch]) (3.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch!=1.12.0,>=1.10->transformers[torch]) (3.1.2)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-
packages (from torch!=1.12.0,>=1.10->transformers[torch]) (2.1.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers[torch])
(3.3.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers[torch]) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers[torch])
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers[torch])
(2023.7.22)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from
jinja2->torch!=1.12.0,>=1.10->transformers[torch]) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch!=1.12.0,>=1.10->transformers[torch]) (1.3.0)
```

[ ]:
```python
import torch
import torch.nn as nn
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import Trainer, TrainingArguments
from transformers import DataCollatorWithPadding
```

[ ]:
```python
class DrugCombDataset(Dataset):
    def __init__(self, df):
        self.df = df

    def __len__(self):
        return len(self.df)
```

```
    def __getitem__(self, index):
        column_names = [
            ("drug_row", "The first drug is "),
            ("drug_col", ". The second drug is "),
            ("cell_line_name", ". Cell line is "),
            ("tissue_name", ". Tissue is "),
            ("ri_row", ". First drug's sensitivity score using relative␣
↪inhibition is "),
            ("ri_col", ". Second drug's sensitivity score using relative␣
↪inhibition is ")
        ]

        # synergy = df.iloc[index]["synergy_class"]

        x_strs = [f"{col_desc}{self.df.iloc[index][col]}" for col, col_desc in␣
↪column_names]
        # random.shuffle(x_strs)
        x_str = ''.join(x_strs)
        x_str = x_str.replace('\n', '')
        x_str = 'Decide in a single word if the synergy of the drug combination␣
↪in the cell line is bad or good. '+x_str
        x_str = x_str+'. Synergy:'
        # print(x_str)

        #tokens = tokenizer(x_str, max_length=128, padding='max_length',␣
↪truncation=True, return_tensors="pt")
        tokens = tokenizer(x_str, return_tensors="pt")
        item = { k: v[0] for k, v in tokens.items() }
        item['labels'] = torch.tensor(self.df.iloc[index]['synergy_class'])

        return item
```

load pretrained cancerGPT (cancerGPT finetuned on GPT2 using common cancer drug synergy combination dataset)

[ ]:
```
id2label = {0: "bad", 1: "good"}
label2id = {"bad": 0, "good": 1}

MODEL_NAME = 'drive/MyDrive/AIHealthTutorial/Online/LLM/cancergpt.pt'
tokenizer = AutoTokenizer.from_pretrained('gpt2')
tokenizer.padding_side = 'left'
tokenizer.pad_token = tokenizer.eos_token
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
model = AutoModelForSequenceClassification.from_pretrained('gpt2', num_labels =␣
↪2, id2label=id2label, label2id=label2id)
model.load_state_dict(torch.load(MODEL_NAME, map_location=torch.device('cpu')))
model.resize_token_embeddings(len(tokenizer))
```

```
model.config.pad_token_id = model.config.eos_token_id
```

```
Some weights of GPT2ForSequenceClassification were not initialized from the
model checkpoint at gpt2 and are newly initialized: ['score.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[ ]:
```
# taking the pre-defined test set about endometrium which is same as the above␣
 ↪endometrium test set
test_indices = data_split['endometrium']['test']
test_df = df.iloc[test_indices]
test_ds = DrugCombDataset(test_df)
```

[ ]:
```
training_args = TrainingArguments(
    output_dir ='results_TabLLM_classification/test',
    num_train_epochs = 10,
    per_device_train_batch_size = 64,
    per_device_eval_batch_size = 64,
    weight_decay = 0.01,
    learning_rate = 5e-4,
    logging_dir = 'logs',
    save_total_limit = 10,
    load_best_model_at_end = False,
    evaluation_strategy = "no",
    save_strategy = "no",
    logging_steps=1,
    report_to=None,
)

def compute_metrics(eval_pred):

    predictions, labels = eval_pred
    return {'AUROC':roc_auc_score(labels, predictions[:,1]),'AUPRC':
 ↪average_precision_score(labels, predictions[:,1])}

trainer = Trainer(
    model = model,
    args = training_args,
    train_dataset = test_ds,
    eval_dataset = test_ds,
    compute_metrics = compute_metrics,
    tokenizer = tokenizer,
    data_collator=data_collator,
)
eva = trainer.evaluate()
auroc = eva['eval_AUROC']
auprc = eva['eval_AUPRC']
```

```
print('\nAUROC:', auroc, '\nAUPRC', auprc)
```

```
<IPython.core.display.HTML object>


AUROC: 0.7959183673469388
AUPRC 0.8468614718614718
```