**人工智慧期末作業**

**利用OPENAI EMBEDDING 進行 自然語言處理應用**

**安裝套件**

**1 openai**

**2transfomers**

**3tikitoken**

**\*\*輸入**

!pip install openai -U

!pip install transformers -U

!pip install tiktoken

引入需要的文件

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/

Collecting openai

Downloading openai-0.27.6-py3-none-any.whl (71 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.9/71.9 kB 391.6 kB/s eta 0:00:00

Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from openai) (2.27.1)

Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai) (4.65.0)

Collecting aiohttp (from openai)

Downloading aiohttp-3.8.4-cp310-cp310-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (1.0 MB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 35.4 MB/s eta 0:00:00

Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (1.26.15)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2022.12.7)

Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2.0.12)

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: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (23.1.0)

Collecting multidict<7.0,>=4.5 (from aiohttp->openai)

Downloading multidict-6.0.4-cp310-cp310-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (114 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.5/114.5 kB 4.2 MB/s eta 0:00:00

Collecting async-timeout<5.0,>=4.0.0a3 (from aiohttp->openai)

Downloading async\_timeout-4.0.2-py3-none-any.whl (5.8 kB)

Collecting yarl<2.0,>=1.0 (from aiohttp->openai)

Downloading yarl-1.9.2-cp310-cp310-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (268 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 11.5 MB/s eta 0:00:00

Collecting frozenlist>=1.1.1 (from aiohttp->openai)

Downloading frozenlist-1.3.3-cp310-cp310-manylinux\_2\_5\_x86\_64.manylinux1\_x86\_64.manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (149 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 149.6/149.6 kB 11.5 MB/s eta 0:00:00

Collecting aiosignal>=1.1.2 (from aiohttp->openai)

Downloading aiosignal-1.3.1-py3-none-any.whl (7.6 kB)

Installing collected packages: multidict, frozenlist, async-timeout, yarl, aiosignal, aiohttp, openai

Successfully installed aiohttp-3.8.4 aiosignal-1.3.1 async-timeout-4.0.2 frozenlist-1.3.3 multidict-6.0.4 openai-0.27.6 yarl-1.9.2

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/

Collecting transformers

Downloading transformers-4.29.2-py3-none-any.whl (7.1 MB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.1/7.1 MB 38.0 MB/s eta 0:00:00

Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.0)

Collecting huggingface-hub<1.0,>=0.14.1 (from transformers)

Downloading huggingface\_hub-0.14.1-py3-none-any.whl (224 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.5/224.5 kB 19.8 MB/s eta 0:00:00

Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.22.4)

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.1)

Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0)

Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2022.10.31)

Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.27.1)

Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)

Downloading tokenizers-0.13.3-cp310-cp310-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (7.8 MB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 75.0 MB/s eta 0:00:00

Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.65.0)

Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers) (2023.4.0)

Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers) (4.5.0)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (1.26.15)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2022.12.7)

Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.12)

Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)

Installing collected packages: tokenizers, huggingface-hub, transformers

Successfully installed huggingface-hub-0.14.1 tokenizers-0.13.3 transformers-4.29.2

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/

Collecting tiktoken

Downloading tiktoken-0.4.0-cp310-cp310-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (1.7 MB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 45.4 MB/s eta 0:00:00

Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2022.10.31)

Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2.27.1)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (1.26.15)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (2022.12.7)

Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (2.0.12)

Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (3.4)

Installing collected packages: tiktoken

Successfully installed tiktoken-0.4.0

\*\*輸入

import pandas as pd

import tiktoken

from openai.embeddings\_utils import get\_embedding

import openai

使用get\_embedding將文字資料轉成文字嵌入

選擇embedding模型

使用最新ada

Encoding選用tokenizer

Token值設為8000 預設最大8191

\*\*輸入

*# embedding model parameters*

embedding\_model = "text-embedding-ada-002"

embedding\_encoding = "cl100k\_base" *# this the encoding for text-embedding-ada-002*

max\_tokens = 8000 *# the maximum for text-embedding-ada-002 is 8191*

上傳資料並合併和評論

1由於上傳比數太大 只用前1000筆資料

2資料從csv讀做df

3dropna將空白欄位移除

4新增欄位將sumarry和text合併為combind

\*\*輸入

!wget <https://github.com/shhuangmust/AI/raw/111-2/fine_food_reviews_1k.csv>

--2023-05-18 11:59:07-- https://github.com/shhuangmust/AI/raw/111-2/fine\_food\_reviews\_1k.csv

Resolving github.com (github.com)... 140.82.121.4

Connecting to github.com (github.com)|140.82.121.4|:443... connected.

HTTP request sent, awaiting response... 302 Found

Location: https://raw.githubusercontent.com/shhuangmust/AI/111-2/fine\_food\_reviews\_1k.csv [following]

--2023-05-18 11:59:07-- https://raw.githubusercontent.com/shhuangmust/AI/111-2/fine\_food\_reviews\_1k.csv

Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...

Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 436148 (426K) [text/plain]

Saving to: ‘fine\_food\_reviews\_1k.csv’

fine\_food\_reviews\_1 100%[===================>] 425.93K --.-KB/s in 0.01s

2023-05-18 11:59:07 (33.3 MB/s) - ‘fine\_food\_reviews\_1k.csv’ saved [436148/436148]

\*\*輸入

input\_datapath = "fine\_food\_reviews\_1k.csv" *# to save space, we provide a pre-filtered dataset*

df = pd.read\_csv(input\_datapath, index\_col=0)

df = df[["Time", "ProductId", "UserId", "Score", "Summary", "Text"]]

df = df.dropna()

df["combined"] = (

"Title: " + df.Summary.str.strip() + "; Content: " + df.Text.str.strip()

)

df.head(2)

一張含有 文字, 字型, 行, 螢幕擷取畫面 的圖片

自動產生的描述

只取前1000個 並將太長文字去掉

1取得前1000個

2 drop較舊的

3tiktoken將文字encodind城需要的

4trncate太長的

\*\*輸入

*# subsample to 1k most recent reviews and remove samples that are too long*

top\_n = 1000

df = df.sort\_values("Time").tail(top\_n \* 2) *# first cut to first 2k entries, assuming less than half will be filtered out*

df.drop("Time", axis=1, inplace=True)

encoding = tiktoken.get\_encoding(embedding\_encoding)

*# omit reviews that are too long to embed*

df["n\_tokens"] = df.combined.apply(lambda x: len(encoding.encode(x)))

df = df[df.n\_tokens <= max\_tokens].tail(top\_n)

len(df)

輸出

1000

\*\*輸入

openai.api\_key = "請輸入你自己的api-key"

\*\*輸入

openai.api\_key = "請輸入你自己的api-key"

\*\*輸入

*# 如果沒有api key，上一格程式不要執行，改執行下面wget命令*

*#!wget* [*https://github.com/joshhu/mustgpt2023/raw/main/Code/5/fine\_food\_reviews\_with\_embeddings\_1k.csv*](https://github.com/joshhu/mustgpt2023/raw/main/Code/5/fine_food_reviews_with_embeddings_1k.csv)

結果

1先降維

\*\*輸入

*# 如果沒有api key，上一格程式不要執行，改執行下面wget命令*

*#!wget* [*https://github.com/joshhu/mustgpt2023/raw/main/Code/5/fine\_food\_reviews\_with\_embeddings\_1k.csv*](https://github.com/joshhu/mustgpt2023/raw/main/Code/5/fine_food_reviews_with_embeddings_1k.csv)

\*\*輸出

(1000, 2)

會將結果繪製出來 在畫出來

\*\*輸入

import matplotlib.pyplot as plt

import matplotlib

import numpy as np

colors = ["red", "darkorange", "gold", "turquoise", "darkgreen"]

x = [x for x,y in vis\_dims]

y = [y for x,y in vis\_dims]

color\_indices = df.Score.values - 1

colormap = matplotlib.colors.ListedColormap(colors)

plt.scatter(x, y, c=color\_indices, cmap=colormap, alpha=0.3)

for score in [0,1,2,3,4]:

avg\_x = np.array(x)[df.Score-1==score].mean()

avg\_y = np.array(y)[df.Score-1==score].mean()

color = colors[score]

plt.scatter(avg\_x, avg\_y, marker='x', color=color, s=100)

plt.title("Amazon ratings visualized in language using t-SNE")

輸出

Text(0.5, 1.0, 'Amazon ratings visualized in language using t-SNE')

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

* ##**如不想執行，可直接下載檔案 (改執行下面的wget命令，請把#註解拿掉)**

\*\*輸入

def get\_embedding2(text, model="text-embedding-ada-002"):

text = text.replace("\n", " ")

return openai.Embedding.create(input = [text], model=model)['data'][0]['embedding']

df['ada\_embedding'] = df.combined.apply(lambda x: get\_embedding2(x, model='text-embedding-ada-002'))

df.to\_csv('embedded\_1k\_reviews.csv', index=False)

\*\*輸入

*# 如果不想執行上一格程式，可改執行下面wget命令*

*#!wget* [*https://github.com/joshhu/mustgpt2023/raw/main/Code/5/embedded\_1k\_reviews.csv*](https://github.com/joshhu/mustgpt2023/raw/main/Code/5/embedded_1k_reviews.csv)

#用新的程式 重畫一個圖

\*\*輸入

import pandas as pd

from sklearn.manifold import TSNE

import matplotlib.pyplot as plt

import matplotlib

df = pd.read\_csv('embedded\_1k\_reviews.csv')

matrix = np.array(df.ada\_embedding.apply(eval).to\_list())

*# Create a t-SNE model and transform the data*

tsne = TSNE(n\_components=2, perplexity=15, random\_state=42, init='random', learning\_rate=200)

vis\_dims = tsne.fit\_transform(matrix)

colors = ["red", "darkorange", "gold", "turquoise", "darkgreen"]

x = [x for x,y in vis\_dims]

y = [y for x,y in vis\_dims]

color\_indices = df.Score.values - 1

colormap = matplotlib.colors.ListedColormap(colors)

plt.scatter(x, y, c=color\_indices, cmap=colormap, alpha=0.3)

for score in [0,1,2,3,4]:

avg\_x = np.array(x)[df.Score-1==score].mean()

avg\_y = np.array(y)[df.Score-1==score].mean()

color = colors[score]

plt.scatter(avg\_x, avg\_y, marker='x', color=color, s=100)

plt.title("Amazon ratings visualized in language using t-SNE")

輸出

Text(0.5, 1.0, 'Amazon ratings visualized in language using t-SNE')

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

用此embedding做regression

\*\*輸入

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

datafile\_path = "fine\_food\_reviews\_with\_embeddings\_1k.csv"

df = pd.read\_csv(datafile\_path)

df["embedding"] = df.embedding.apply(eval).apply(np.array)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(list(df.embedding.values), df.Score, test\_size=0.2, random\_state=42)

rfr = RandomForestRegressor(n\_estimators=100)

rfr.fit(X\_train, y\_train)

preds = rfr.predict(X\_test)

mse = mean\_squared\_error(y\_test, preds)

mae = mean\_absolute\_error(y\_test, preds)

print(f"ada-002 embedding performance on 1k Amazon reviews: mse={mse:.2f}, mae={mae:.2f}")

輸出

ada-002 embedding performance on 1k Amazon reviews: mse=0.60, mae=0.52

\*\*輸入

bmse = mean\_squared\_error(y\_test, np.repeat(y\_test.mean(), len(y\_test)))

bmae = mean\_absolute\_error(y\_test, np.repeat(y\_test.mean(), len(y\_test)))

print(

f"Dummy mean prediction performance on Amazon reviews: mse={bmse:.2f}, mae={bmae:.2f}"

)

輸出

Dummy mean prediction performance on Amazon reviews: mse=1.73, mae=1.03

用此做分類

\*\*輸入

*# imports*

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, accuracy\_score

*# load data*

datafile\_path = "fine\_food\_reviews\_with\_embeddings\_1k.csv"

df = pd.read\_csv(datafile\_path)

df["embedding"] = df.embedding.apply(eval).apply(np.array) *# convert string to array*

*# split data into train and test*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

list(df.embedding.values), df.Score, test\_size=0.2, random\_state=42

)

*# train random forest classifier*

clf = RandomForestClassifier(n\_estimators=100)

clf.fit(X\_train, y\_train)

preds = clf.predict(X\_test)

probas = clf.predict\_proba(X\_test)

report = classification\_report(y\_test, preds)

print(report)

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

繪製圖形

\*\*輸入

from openai.embeddings\_utils import plot\_multiclass\_precision\_recall

plot\_multiclass\_precision\_recall(probas, y\_test, [1, 2, 3, 4, 5], clf)

輸出

RandomForestClassifier() - Average precision score over all classes: 0.88

一張含有 文字, 圖表, 行, 繪圖 的圖片

自動產生的描述

用此embedding做zero shot分析

\*\*輸入

import pandas as pd

import numpy as np

from sklearn.metrics import classification\_report

*# parameters*

EMBEDDING\_MODEL = "text-embedding-ada-002"

*# load data*

datafile\_path = "fine\_food\_reviews\_with\_embeddings\_1k.csv"

df = pd.read\_csv(datafile\_path)

df["embedding"] = df.embedding.apply(eval).apply(np.array)

*# convert 5-star rating to binary sentiment*

df = df[df.Score != 3]

df["sentiment"] = df.Score.replace({1: "negative", 2: "negative", 4: "positive", 5: "positive"})

輸出

一張含有 文字, 螢幕擷取畫面, 字型, 代數 的圖片

自動產生的描述

\*\*輸入

from openai.embeddings\_utils import cosine\_similarity, get\_embedding

from sklearn.metrics import PrecisionRecallDisplay

def evaluate\_embeddings\_approach(

labels = ['negative', 'positive'],

model = EMBEDDING\_MODEL,

):

label\_embeddings = [get\_embedding(label, engine=model) for label in labels]

def label\_score(review\_embedding, label\_embeddings):

return cosine\_similarity(review\_embedding, label\_embeddings[1]) - cosine\_similarity(review\_embedding, label\_embeddings[0])

probas = df["embedding"].apply(lambda x: label\_score(x, label\_embeddings))

preds = probas.apply(lambda x: 'positive' if x>0 else 'negative')

report = classification\_report(df.sentiment, preds)

print(report)

display = PrecisionRecallDisplay.from\_predictions(df.sentiment, probas, pos\_label='positive')

\_ = display.ax\_.set\_title("2-class Precision-Recall curve")

evaluate\_embeddings\_approach(labels=['negative', 'positive'], model=EMBEDDING\_MODEL)

輸出

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述

\*\*輸入

evaluate\_embeddings\_approach(labels=['An Amazon review with a negative sentiment.', 'An Amazon review with a positive

輸出

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 陳列, 數字 的圖片

自動產生的描述

\*\*輸入

evaluate\_embeddings\_approach(labels=['An Amazon review with a negative sentiment.', 'An Amazon review with a positive sentiment.'])

輸出

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 行, 圖表 的圖片

自動產生的描述

**做clustering**

\*\*輸入

*# imports*

import numpy as np

import pandas as pd

*# load data*

datafile\_path = "fine\_food\_reviews\_with\_embeddings\_1k.csv"

df = pd.read\_csv(datafile\_path)

df["embedding"] = df.embedding.apply(eval).apply(np.array) *# convert string to numpy array*

matrix = np.vstack(df.embedding.values)

matrix.shape

輸出

(1000, 1536)

\*\*輸入

from sklearn.cluster import KMeans

n\_clusters = 4

kmeans = KMeans(n\_clusters=n\_clusters, init="k-means++", random\_state=42)

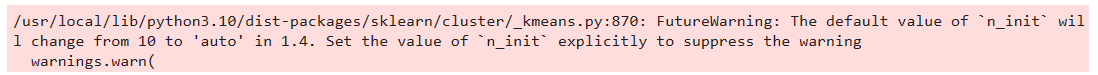
kmeans.fit(matrix)

labels = kmeans.labels\_

df["Cluster"] = labels

df.groupby("Cluster").Score.mean().sort\_values()

輸出



一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

\*\*輸入

from sklearn.manifold import TSNE

import matplotlib

import matplotlib.pyplot as plt

tsne = TSNE(n\_components=2, perplexity=15, random\_state=42, init="random", learning\_rate=200)

vis\_dims2 = tsne.fit\_transform(matrix)

x = [x for x, y in vis\_dims2]

y = [y for x, y in vis\_dims2]

for category, color in enumerate(["purple", "green", "red", "blue"]):

xs = np.array(x)[df.Cluster == category]

ys = np.array(y)[df.Cluster == category]

plt.scatter(xs, ys, color=color, alpha=0.3)

avg\_x = xs.mean()

avg\_y = ys.mean()

plt.scatter(avg\_x, avg\_y, marker="x", color=color, s=100)

plt.title("Clusters identified visualized in language 2d using t-SNE")

輸出

Text(0.5, 1.0, 'Clusters identified visualized in language 2d using t-SNE')

一張含有 螢幕擷取畫面, 鮮豔, 文字 的圖片

自動產生的描述

#用embedding做語意搜尋

\*\*輸入

import pandas as pd

import numpy as np

datafile\_path = "fine\_food\_reviews\_with\_embeddings\_1k.csv"

df = pd.read\_csv(datafile\_path)

df["embedding"] = df.embedding.apply(eval).apply(np.array)

\*\*輸入

from openai.embeddings\_utils import get\_embedding, cosine\_similarity

*# search through the reviews for a specific product*

def search\_reviews(df, product\_description, n=3, pprint=True):

product\_embedding = get\_embedding(

product\_description,

engine="text-embedding-ada-002"

)

df["similarity"] = df.embedding.apply(lambda x: cosine\_similarity(x, product\_embedding))

results = (

df.sort\_values("similarity", ascending=False)

.head(n)

.combined.str.replace("Title: ", "")

.str.replace("; Content:", ": ")

)

if pprint:

for r in results:

print(r[:200])

print()

return results

results = search\_reviews(df, "delicious beans", n=3)

輸出

一張含有 文字, 字型, 螢幕擷取畫面, 代數 的圖片

自動產生的描述

\*\*輸入

results = search\_reviews(df, "whole wheat pasta", n=3)

輸出

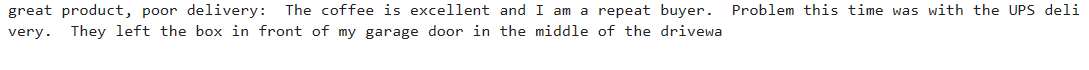
一張含有 文字, 螢幕擷取畫面, 字型, 代數 的圖片

自動產生的描述

\*\*輸入

results = search\_reviews(df, "bad delivery", n=1)

輸出



\*\*輸入

results = search\_reviews(df, "spoilt", n=1)

輸出



\*\*輸入

results = search\_reviews(df, "pet food", n=2)

輸出

