import pandas as pd

import openai

import json

import ast

openai.api\_base = “ ”

openai.api\_key = “ ”

openai.api\_version = “2023-05-15”

openai.api\_type = “azure”

from langchain.chains.openai\_functions import create\_openai\_fn\_chian

from langchain.chat\_models import AzureChatOpenAI, ChatOpenAI

from langchain.prompts import SystemMessagePromptTemplate, HumanMessagePromptTemplate, ChatPromptTemplate

from langchain.pydantic\_v1 import BaseModel, Field

# Sometiems GPT’s outputs are not json type, this function is to check this situation.

def is\_json\_valid(var):

try:

json.loads(var)

return True

except json.JSONDecodeError:

return False

def grading\_function(analysis\_prompt, resume\_content):

“””

analysis\_prompt is written for GPT, resume\_content is each candidate’s resume.

This funciton combines the prompt and the resume for grading.

“””

# First Step: Using GPT to gain 5 info: whether the academy meets the requirement, three scores, reason.

analysis\_system\_message\_prompt = SystemMessagePromptTemplate.from\_template(

template = analysis\_prompt

)

candidate\_prompt = “””CV: {text}”””

analysis\_human\_message\_prompt = HumanMessagePromptTemplate.from\_template(

candidate\_prompt

)

analysis\_chat\_prompt = ChatPromptTemplate.from\_messages(

[analysis\_system\_message\_prompt, analysis\_human\_message\_prompt]

)

first\_messages=analysis\_chat\_prompt.format\_prompt(text=resume\_content).to\_messages()

model = AzureChatOpenAI(

Openai\_api\_version=”2023-05-15”,

Temperature=0,

Request\_timeout=60,

Max\_tokens=1024,

)

token\_len = model.get\_num\_tokens\_from\_messages(fisrt\_messages)

model.deployment\_name = “gpt-35-turbo”if token\_len < 2000 else “gpt-35-turbo-16k”

prompt = str(model(first\_messages))

#Second Step: Select 5 info from GPT’s output,. Using GPT to add three scores as the candidate’s final grade.

second\_messages = [{“role”: “user”, “content”: prompt}]

function = [

{

“name”: “get\_info”,

“description”: “Get info: whether the candidate pass the test, three scores, and reason for candidate”,

“parameters”: {

“type”: “object”,

“properties”: {

“Pass”: {“type”: “string”, “description”: “whether the candidate pass the test, according to their academic degree. ”},

“score\_exp”: {

“type”: “integer”,

“description”: “The experience score of the candidate”,

},

“score\_academy”: {

“type”: “integer”,

“description”: “The academy score of the candidate”,

},

“score\_award”: {

“type”: “integer”,

“description”: “The award score of the candidate”,

},

“analysis”: {“type”: “string”, “description”: “The analysis from three scores mentioned above. ”},

“summary”: {“type”: “string”, “description”: “A short summary of the analysis. ”},

},

“required”: [“Pass, score\_exp, score\_academy, score\_award, analysis, summary”],

},

}

]

openai.api\_base = “https: ”

openai.api\_key = “ ”

openai.api\_version = “2023-07-01-preview”

openai.api\_type = “azure”

second\_response = openai.ChatCompletion.create(

engine = “gpt-35-turbo”,

messages = second\_messages,

functions = functions,

function\_call = “auto”,

)

response\_message = second\_response[“choices”][0][“message”]

# Third Step: check whether the second step’s output is json type.

arguments = response\_message[“function\_call”][“arguments”]

If is\_json\_valid(arguments):

arguments = json.loads(arguments)

whether\_pass = arguments[“Pass”]

score\_exp = arguments[“score\_exp”]

score\_academy = arguments[“score\_academy”]

score\_award = arguments[“score\_award”]

analysis = arguments[“analysis”]

summary = arguments[“summary”]

else:

whether\_pass = str(“数值丢失”)

score\_exp = -1

score\_academy = -1

score\_award = -1

analysis = str(“数值丢失”)

summary = str(“数值丢失”)

return whether\_pass, score\_exp, score\_academy, score\_award, analysis, summary

# 编写prompt，对于每一份简历，判断学历是否符合要求，根据学历，工作、实习经历，和获奖情况打分，并给出打分理由

analysis\_prompt = f”””

You are a professional human resource who work for Luckin Coffee. You need to grade each candidate according to their resume. \

You also need to judge whether their experiences is related to catering industry, coffee, tea, milk tea, yogurt, or waiter. \

df varaiable contains all the resume. \

In the resume, some information is written in English, the rest of the information is written in Chinese. \

You need to focus in both English and Chinese information. \

Judge Experience Method:

1 - The number of project or experience is 0, score\_exp = 0, \

2 - The candidate has project or experience, neither of them is directly related to food or beverage area or waiter. \

3 - The candidate has project or experience, among them, one of them is directly related to food or beverage area or waiter. \

4 - The candidate has project or experience, among them, two of them is directly related to food or beverage area or waiter. \

5 - The candidate has project or experience, among them, more than two of them is directly related to food or beverage area or waiter. \

Judge Academy Method:

1 - academicDegree is “高中” or “中专”， score\_academy = 10. \

2 - academicDegree is “大专”， score\_academy = 15. \

3 - academicDegree is “本科” or “硕士” or “博士”， score\_academy = 20. \

Judge Award Method:

1 - The number of award is 0, score\_award = 0. \

2 - The number of award is equal or larger than 0, score\_award = 15. \

Think step by step:

1 - Judging whether the candidate pass the test: \

if the academicDegree is “专科” or “本科”, then the candidate pass the test. \

if the academicDegree is “高中” or “中专”, then the candidate doesn’t pass the test. \

2 - Print whether the candidate pass the test or not. \

3 - The initial value of score\_exp is 0. \

Using <experienceInfo> in the resume, obtain score\_exp and remember this number, \

detailed scoring criteria seen in <Judge Experience Method>. \

4 - The initial value of score\_academy is 0. \

Using <academicDegree> in the resume, obtain score\_academy and remember this number, \

detailed scoring criteria seen in <Judge Academy Method>. \

5 - The initial value of score\_award is 0. \

Using <awardInfo> in the resume, obtain score\_award and remember this number, \

detailed scoring criteria seen in <Judge Award Method>. \

6 - The score\_exp, score\_academy, and score\_award do not affect each other. \

7 - Print the score\_exp, score\_academy, and score\_award. \

8 - Give a detailed analysis from experience, academy, and award perspectives, \

List the experience that related to the food or beverage area(if the candidate has), \

double check whether it is related to the food or beverage area. \

9 - Give a summary of the analysis mentioned below. \

Format:

1. Pass: Yes/No.
2. score\_exp, score\_academy, score\_award.
3. Analysis.
4. Summary.

“””

#编写prompt，将每份简历得到的英文打分理由翻译为中文

def translation(prompt\_reason, reason):

analysis\_system\_message\_prompt = SystemMessagePromptTemplate.from\_template(

template = prompt\_reason

)

reason\_prompt = “””Input: {reason}”””

Analysis\_human\_message\_prompt = HumanMessagePromptTemplate.from\_template(

Reason\_prompt

)

analysis\_chat\_prompt = ChatPromptTemplate.from\_messages(

[analysis\_system\_message\_prompt, analysis\_human]

)

trans\_messages = analysis\_chat\_prompt.format\_prompt(reason=reason).to\_messages()

model = AzureChatOpenAI(

openai\_api\_version = “2023-05-15”,

temperature = 0,

request\_timeout = 60,

max\_token = 1024,

)

token\_len = model.get\_nun\_tokens\_from\_messages(trans\_messages)

model.deployment\_name = “gpt-35-turbo” if token\_len <2000 else “gpt-35-turbo-16k”

return str(model(trans\_messages).content)

Prompt\_chinese = f”””

You are a professional translator.

Your job is to translate the reason from English to elegant Chinese.

Reason: ```<reason>```

“””

#main function

def full\_time(df):

df\_new = df[[‘dt’, ‘applicationId’, ‘candidateId’, ‘company’, ‘position\_title’, ‘resume\_content’]].copy()

is\_recommended = []

total\_score = []

recommend\_analysis = []

recommend\_reason = []

summary = []

for i in range(0, len(df)):

text = df.iloc[i, -3].replace(“nan”, “0”)

text = ast.literal\_eval(text)

text.pop(0)

text.pop(0)

text.pop(1)

is\_recommended\_temp, score\_exp\_temp, score\_academy\_temp, score\_award\_temp, analysis\_temp, summary\_temp = grading\_function(analysis\_prompt, text)

total\_score\_temp = score\_exp\_temp + score\_academy\_temp + score\_award\_temp

#Check if the grade overflows.

If total\_score\_temp > 100:

Total\_score\_temp = 100

analysis\_chinese\_temp = translation(prompt\_chinese, analysis\_temp)

summary\_chinese\_temp = translation(prompt\_chinese, summary\_temp)

print(i, “success”, is\_recommended\_temp, analysis\_chinese\_temp)

is\_recommended.append(is\_recommended\_temp)

total\_score.append(total\_score\_temp)

recommend\_analysis.append(analysis\_chinese\_temp)

recommend\_reason.append(analysis\_chinese\_temp)

summary.append(summary\_chinese\_temp)

df\_new[“recommend\_analysis”] = recommend\_analysis

df\_new[“recommend\_reason”] = recommend\_reason

df\_new[“is\_recommended”] = is\_recommended

df\_new[“total\_score”] = total\_score

df\_new[“summary”] = summary

return df\_new

#\*\*\*\*

from fulltime import full\_time

import pandas as pd

df = pd.read\_csv(‘barista\_1103.csv’)

new\_df = full\_time(df)

new\_df.to\_csv(‘final\_1103.csv’)