调用大语言模型完成文本分类任务

本tutorial来自https://github.com/Huang-Jingxiang/call_llm4classification ，仅供学习交流使用，欢迎star🌟 。

前言

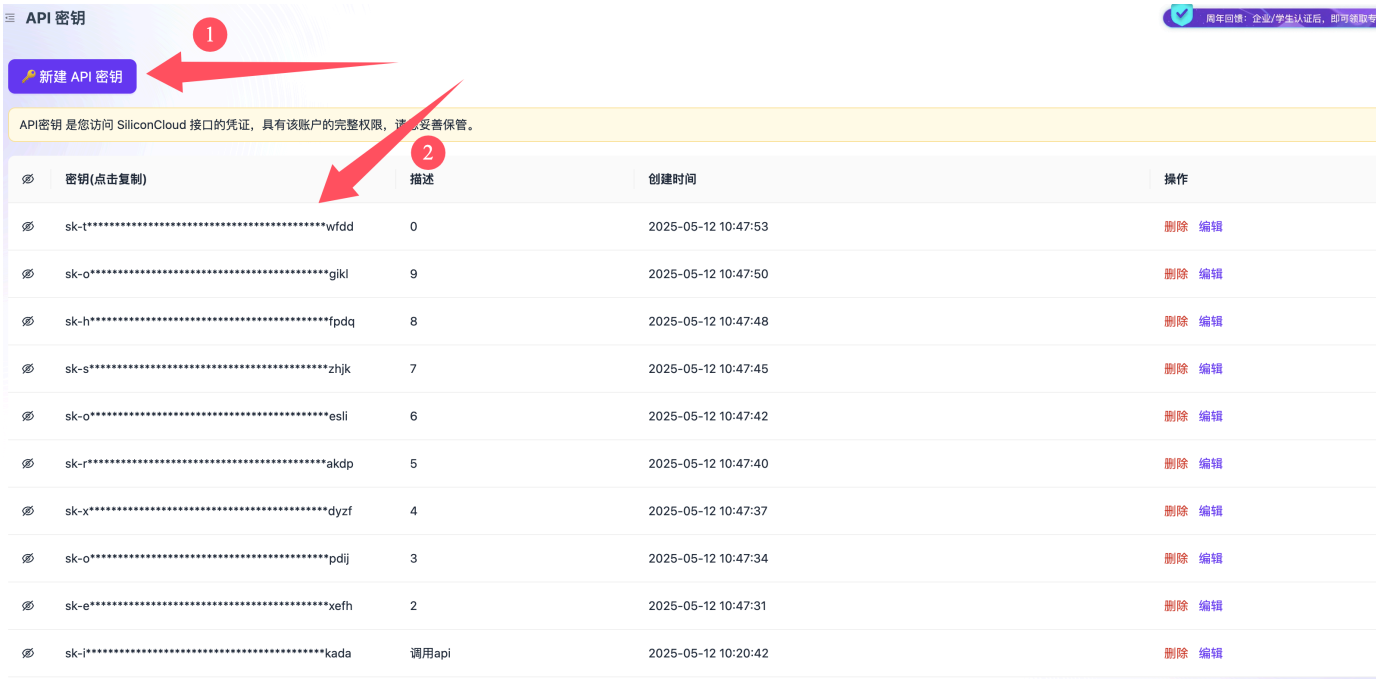
- 自从2022年11月30日ChatGPT问世以来，大语言模型的应用范围越来越广泛，为我们的学习生活带来越来越多的便利。本tutorial将介绍如何调用大语言模型完成分类任务。
- 何谓分类任务?
  - 分类任务是指将输入的文本数据分为事先指定好的不同的类别。比如，我们可以将新闻文章按主题分为体育、娱乐、政治等类别；将社交媒体上的博文按情感分为正面、负面、中性等类别……
  - 在大语言模型问世前，分类任务主要由传统的机器学习模型完成，比如支持向量机（SVM）、朴素贝叶斯（Naive Bayes）、决策树（Decision Tree）等。这些模型需要先对文本进行特征提取，然后根据提取的特征进行分类，再训练模型等等，实现上相对来说是较麻烦的，对代码能力的要求也比较高。
  - 但是，大语言模型的出现改变了这一状况。一方面，因为大语言模型具备丰富的预训练知识，所以在完成分类任务时，提取特征-训练模型的过程并非必须，我们只需要调用大语言模型的接口，输入文本数据，即可得到分类结果。另一方面，相对编写完整的训练机器学习的代码而言，调用大语言模型接口的代码更轻量化，实现起来更简单。

实验环境要求

- Python 3.8（只是测试的时候使用的py3.8，理论上更高版本也不会冲突）
- requests、pandas、tqdm、sklearn

如何获得API Key

- 访问SiliconFlow，注册账号并登录。
- 点击“我的”-“API Key”，即可获得API Key。



第一次调用：以代码的形式调用大语言模型

调用的示例代码如下，使用的是SiliconFlow的免费模型 `THUDM/GLM-Z1-9B-0414` ，如果运行代码后能够看到返回的结果，说明调用成功。

```
import requests

url = "https://api.siliconflow.cn/v1/chat/completions"

payload = {
    "model": "THUDM/GLM-Z1-9B-0414",
    "messages": [
        {
            "role": "user",
            "content": "你好"
        }
    ]
}
headers = {
    "Authorization": "Bearer API_Key", # 这个地方的API_Key替换成你在SiliconFlow上获得的API
    "Content-Type": "application/json"
}

response = requests.post(url, json=payload, headers=headers)

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

返回结果示例：

> 你好！有什么我可以帮助你的吗？😊

第二次调用：根据自己的需求调用大语言模型

在这里，我们要让大模型完成一个简单的文本分类任务，即判断文本是正面还是负面情感。示例代码如下：

```python
import requests
import time

url = "https://api.siliconflow.cn/v1/chat/completions"
headers = {
    "Authorization": "Bearer 你的API Key", # 同上，需要修改
    "Content-Type": "application/json"
}

payload = {
    "model": "THUDM/GLM-Z1-9B-0414",
    "messages": [
        {
            "role": "user",
            "content": """你是一个情感分类专家，善于判断文本的情感倾向。现在，我需要你协助我完成文本
            1. 我会输入一段酒店评论，你需要协助我判断这段酒店评论的情感倾向。
            2. 情感倾向分为积极情感与消极情感。
            3. 如果你认为这段评论属于积极情感，则返回1；如果属于消极情感，则返回0.
            4. 不要输出任何其他内容，只输出0或1.

            input：环境交通都不错，房间也可以，下次还会再来
            output：
            """
        }
    ]
}

def get_sentiment_result(max_retries=5):
    retries = 0
    while retries < max_retries:
        try:
            response = requests.post(url, json=payload, headers=headers)
            response.raise_for_status()
            content = response.json()['choices'][0]['message']['content'].strip()

            # 清洗结果，仅保留0或1
            clean_output = content.strip().replace("\n", "").replace(" ", "")
            if clean_output in ["0", "1", 0, 1]:
                return clean_output
            else:
                print(f"⚠️ 无效输出：{content}，重新请求中...（第 {retries+1} 次）")
                retries += 1
                time.sleep(1)  # 避免频繁请求
        except Exception as e:
            print(f"❌ 请求或解析出错：{e}，重新尝试中...（第 {retries+1} 次）")
            retries += 1
```

```
            time.sleep(1)

    raise RuntimeError("多次请求仍未获得有效结果（0或1）")

# 调用函数获取结果
result = get_sentiment_result()
print(result)
```

正常运行后，应该能得到输出1。
相较于上面的代码，这个代码的不同之处在于：

1. 我们在prompt中添加了情感分类的任务描述，即要求大模型根据任务描述进行情感分类。
2. 我们在get_sentiment_result函数中添加了一个清洗结果的步骤，即去掉输出中的空格、换行符等，只保留0或1。
3. 我们在get_sentiment_result函数中添加了一个异常处理的步骤，即当请求或解析出错时（如调用速率到达上限、输出了异常内容），会重新尝试请求，最多重试5次。

第三次调用：从本地读取数据，完成批量情感分类

现实情况下，我们待分类的文本往往是存放在文件中的，这部分的代码则演示了如何从文件中读取数据，完成批量情感分类。

```
import requests
import time
import pandas as pd
from tqdm import tqdm
import json

# LLM API配置
url = "https://api.siliconflow.cn/v1/chat/completions"
headers = {
    "Authorization": "Bearer 你的API Key",  # 替换为你的真实 API key
    "Content-Type": "application/json"
}

def get_sentiment_result(review_text, max_retries=5):
    """调用LLM判断单条评论情感"""
    base_prompt = f"""你是一个情感分类专家，善于判断文本的情感倾向。现在，我需要你协助我完成文本情感
1. 我会输入一段酒店评论，你需要协助我判断这段酒店评论的情感倾向。
2. 情感倾向分为积极情感与消极情感。
3. 如果你认为这段评论属于积极情感，则返回1；如果属于消极情感，则返回0.
4. 不要输出任何其他内容，只输出0或1.

input: {review_text}
output:
"""

    payload = {
        "model": "THUDM/GLM-Z1-9B-0414",
        "messages": [
            {"role": "user", "content": base_prompt}
        ]
    }

    retries = 0
    while retries < max_retries:
        try:
            payload_json = json.dumps(payload, ensure_ascii=False).encode('utf-8')
            response = requests.post(url, data=payload_json, headers=headers)
            response.raise_for_status()
            content = response.json()['choices'][0]['message']['content'].strip()
            clean_output = content.replace("\n", "").replace(" ", "").strip()
            if clean_output in ["0", "1"]:
                return int(clean_output)
            else:
                print(f"⚠️ 非法输出: {clean_output}, 第 {retries + 1} 次重试...")
                retries += 1
                time.sleep(1)
```

```
        except Exception as e:
            print(f"❌ 异常：{e}，第 {retries + 1} 次重试...")
            retries += 1
            time.sleep(1)

    return None  # 如果重试失败，返回空值

# 批量情感分析 + tqdm进度条
results = []
for i, row in tqdm(df.iterrows(), total=len(df), desc="情感分析中"):
    review = row['review']
    sentiment = get_sentiment_result(review)
    results.append(sentiment)

# 添加结果到DataFrame
df['llm_output'] = results

# 保存结果
df.to_csv('data_singlethread.csv', index=False)
df.head()
```

当看到df中多了一列名为llm_output的字段就意味着调用成功。可以看到，完成100条数据的情感分类花费了9分钟。

```
情感分析中: 100%|██████████| 100/100 [09:01<00:00,  5.41s/it]
```

	# label	review	# llm_output
0	1	房间不大，但相当舒适．距离热闹的	1
1	1	酒店就是离市区太远。离新国际展会	0
2	1	帮同事订过很多次，酒店位置很好，	1
3	0	就值5元的浴盐瓶要赔40元,这不是主	0
4	0	帮朋友订的，反映不是太好，想必也	0

第四次调用：多线程调用大模型，提升调用效率

上面的代码虽然能够成功完成文本分类任务，但是处理100条数据足足花了9分钟，这有些难以让人接受。这是因为我们每次调用大模型的API时，都需要等待响应时间，而等待的时间随着文本的长度而增加。为了提升效率，我们可以使用多线程的方式，同时调用多个API Key，来并行处理文本分类任务。

```
import requests
import time
import pandas as pd
from tqdm import tqdm
import json
from concurrent.futures import ThreadPoolExecutor, as_completed

url = "https://api.siliconflow.cn/v1/chat/completions"

api_keys = [
    "sk-123",
    "sk-xxxxx2",
    "sk-xxxxx3",
    "sk-xxxxx4",
    "sk-xxxxx5"
] # 在Silicon Flow中生成5个api key，并填入列表api_keys中。例如，第一个api key为"sk-123"，如图所

def get_sentiment_result(review_text, api_key, max_retries=5):
    base_prompt = f"""你是一个情感分类专家，善于判断文本的情感倾向。现在，我需要你协助我完成文本情感
1. 我会输入一段酒店评论，你需要协助我判断这段酒店评论的情感倾向。
2. 情感倾向分为积极情感与消极情感。
3. 如果你认为这段评论属于积极情感，则返回1；如果属于消极情感，则返回0.
4. 不要输出任何其他内容，只输出0或1.

input: {review_text}
output:
"""

    headers = {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

    payload = {
        "model": "THUDM/GLM-Z1-9B-0414",
        "messages": [
            {"role": "user", "content": base_prompt}
        ]
    }

    retries = 0
    while retries < max_retries:
        try:
            payload_json = json.dumps(payload, ensure_ascii=False).encode('utf-8')
            response = requests.post(url, data=payload_json, headers=headers)
            response.raise_for_status()
```

```
            content = response.json()['choices'][0]['message']['content'].strip()
            clean_output = content.replace("\n", "").replace(" ", "").strip()
            if clean_output in ["0", "1"]:
                return int(clean_output)
            else:
                print(f"⚠️ 非法输出: {clean_output}, 第 {retries + 1} 次重试...")
                retries += 1
                time.sleep(1)
        except Exception as e:
            print(f"❌ 异常: {e}, 第 {retries + 1} 次重试...")
            retries += 1
            time.sleep(1)

    return None

# 线程池最大5线程
max_workers = 5

results = [None] * len(df)

def task(idx, review):
    api_key = api_keys[idx % len(api_keys)]  # 轮流选择api key
    sentiment = get_sentiment_result(review, api_key)
    return idx, sentiment

with ThreadPoolExecutor(max_workers=max_workers) as executor:
    futures = [executor.submit(task, i, row['review']) for i, row in df.iterrows()]
    for future in tqdm(as_completed(futures), total=len(df), desc="情感分析中"):
        idx, sentiment = future.result()
        results[idx] = sentiment

df['llm_output'] = results
df.to_csv('data_multithread.csv', index=False)
df.head()
```

从输出结果可以看到，使用5个线程只用了1分39秒即可完成100条数据的情感分类，速度提升约5倍。同时值得注意的是，try except机制成功避免了非法输出对我们的结果产生的干扰。

```
情感分析中:  40%|████      | 40/100 [00:45<00:50,  1.20it/s]
⚠️ 非法输出: 根据该月的借款数据，我们可以推测该用户的借款周期大约为：**15天**具体推演过程：1.假设每月借款日固定在某一天（例如每月1号
情感分析中: 100%|██████████| 100/100 [01:38<00:00,  1.02it/s]
```

	# label	review	# llm_output
0	1	房间不大，但相当舒适．距离热闹的	1
1	1	酒店就是离市区太远。离新国际展会	0
2	1	帮同事订过很多次，酒店位置很好，	1
3	0	就值5元的浴盐瓶要赔40元,这不是主	0
4	0	帮朋友订的，反映不是太好，想必也	0

5 rows x 3 cols 10 per page Page 1 of 1

分类效果评估

从本地读入文件，调用sklearn里的分类效果评估指标函数，即可完成评估。

```
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

df_multi = pd.read_csv('data_multithread.csv')
df_single = pd.read_csv('data_singlethread.csv')

def print_metrics(y_true, y_pred, prefix="结果"):
    acc = accuracy_score(y_true, y_pred)
    pre = precision_score(y_true, y_pred, zero_division=0)
    rec = recall_score(y_true, y_pred, zero_division=0)
    f1 = f1_score(y_true, y_pred, zero_division=0)
    print(f"{prefix}:")
    print(f"  Accuracy:  {acc:.4f}")
    print(f"  Precision: {pre:.4f}")
    print(f"  Recall:    {rec:.4f}")
    print(f"  F1 Score:  {f1:.4f}")
    print()

# df_multi
print_metrics(df_multi['label'], df_multi['llm_output'], prefix="多线程结果")

# df_single
print_metrics(df_single['label'], df_single['llm_output'], prefix="单线程结果")
```

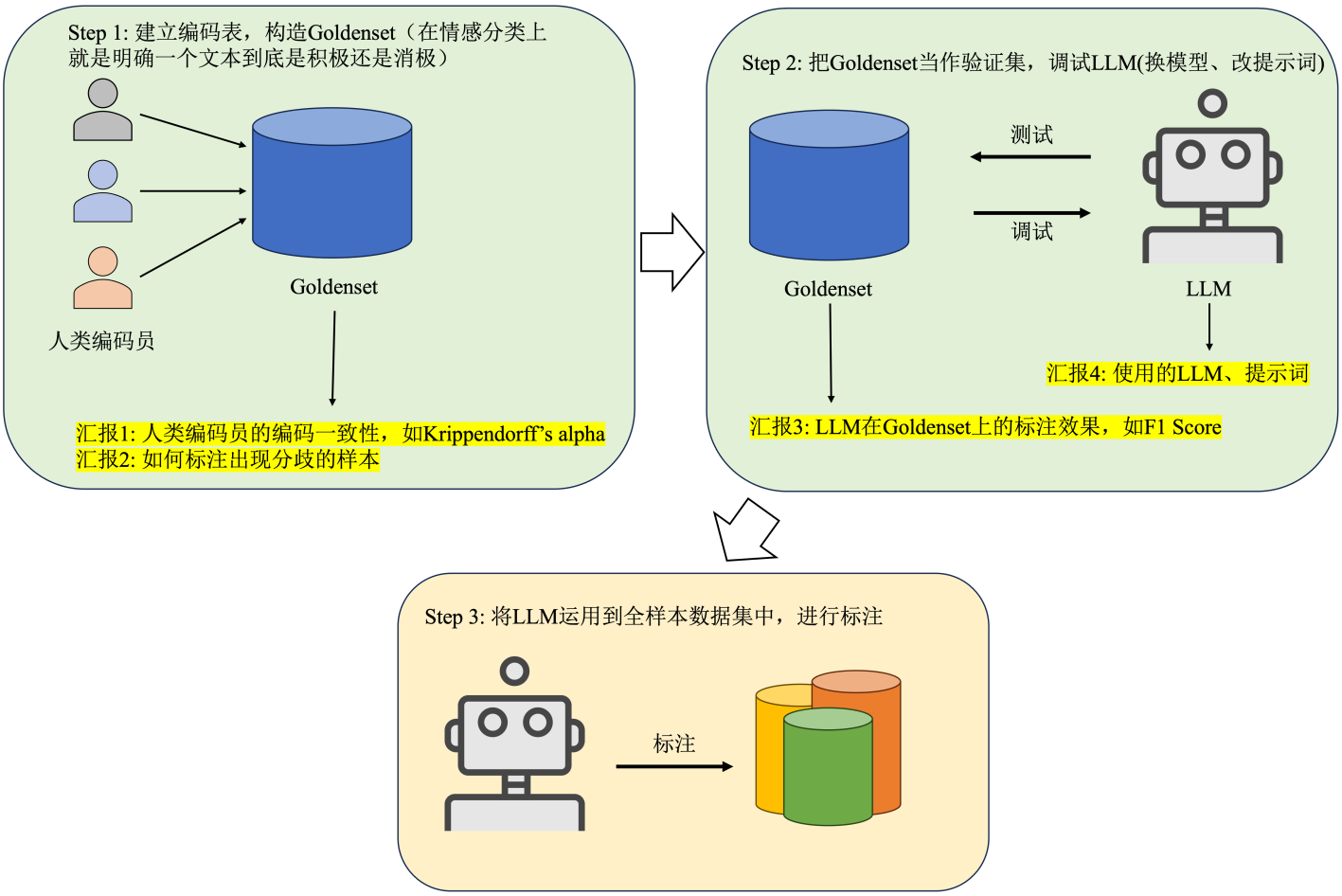
输出结果应如下所示（和tutorial里的数值不一样是很正常的，调api与本地部署模型不一致，没法固定随机种子，所以每次调用进行分类时，分类结果存在不一致的可能性）：

```
多线程结果:
  Accuracy:  0.8400
  Precision: 0.9474
  Recall:    0.7200
  F1 Score:  0.8182

单线程结果:
  Accuracy:  0.8300
  Precision: 1.0000
  Recall:    0.6600
  F1 Score:  0.7952
```

一般来说，F1 Score > 0.8的分类模型，其分类效果在社会科学中就被认为是非常好。评估一个分类模型效果好坏的流程可以表述为下图。具体来说，第一步是建立金标准，在这个案例当中的 `label` 就是一个金标准；第二步，调用模型进行分类，并将模型输出和金标准进行对比，计算评估指标，并根据评估指标来决定是可以进行下一步的大规模测量还是需要重新调整（修改提示词、更换调用的模型）。

LLM可以视作一个不需要进行训练的机器学习模型



后续可以再做些什么?

1. 优化提示词，提高分类效果
2. 尝试Silicon Flow上的其他免费模型，对比同一套提示词下不同模型的分类效果
3. 替换成自己感兴趣的数据，完成除情感分析以外其他领域的分类
4. 了解调用接口时的其他参数的作用，尝试不同的参数，探究参数设置是否会影响最终的结果

再进一步呢?

1. 如果你对大模型分类感兴趣，可以尝试微调大模型来进一步提升分类效果（需要算力资源）
2. 如果你对基于大模型的开发感兴趣，可以试着开发出大模型工作流，完成更复杂的任务（例如，Dify、Langchain等框架都是可以尝试的）

常见问题及解决方案

1. no module named xxx

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[20], line 4
      2 import time
      3 import pandas as pd
----> 4 from tqdm import tqdm
      6 # LLM API配置
      7 url = "https://api.siliconflow.cn/v1/chat/completions"

ModuleNotFoundError: No module named 'tqdm'
```

问题描述：你当前的python环境里缺少某个必要的库，如上图就缺少了显示进度条的库 `tqdm` 。

解决方案：在当前python环境里安装 `tqdm` 库，命令如下：

```
pip install tqdm
```