23 春 Python 与深度学习基础第一次作业

爬虫抓取并可视化不同国家的经济数据

学生姓名：杨益

学号： PB21000308

课程名称

授课教师

中国科学技术大学

May 8, 2023

1 任务描述

在充分考虑鲁棒性的前提下，在 GUI 界面输入年份和一个或多个国家的信息，使用 requests 抓取世界银行[1]公开的 GDP、人口和人均可支配收入数据，绘制折线图，利用 tkinter 进行可视化。其中项目的代码公开在 $https://github.com/YiYang-github/python-and-deep-learning-basics/tree/master$

2 实验细节

本次任务大体分为三个步骤，依次为调用 API 抓取数据 -> 处理数据并绘图 -> 利用 tkinter 进行 GUI 可视化，python 程序主体封装在 class GUI 这个类中。

2.1 调用 API 抓取数据

这一部分对应 class 类中的 get_data 函数，我们先使用.split(",") 把输入的字符串分割为国名的字符串 list 类存储，循环调用 API 读取各国家指定时间段的数据。在调用 API 输入国家、年份信息时，我参考了世界银行 API[2]的提示，API 允许的开始年份需要大于 1960，结束年份小于 2021；而对于国家[3]采取的是" ISO 3166-1 alpha-3 "代码格式，这些在 Readme.md 文档里介绍用法、举例，并且在 main.py 也有提及。

在具体编程程序时，我们先设置一个 $base\_url$ 的字符串，然后填入对应的经济指标和国家名称，爬虫核心代码如下所示。

```
base_url = "https://api.worldbank.org/v2/country"
data_type = {"GDP": "NY.GDP.MKTP.CD", "Population": "SP.POP.TOTL", "GNI per capita":\
"NY.GNP.PCAP.CD"}
...
for country in countries:
        country_data = []
        for key in data_type:
            url = f"{base_url}/{country}/indicator/{data_type[key]}?\
            format=json&per_page=1000&date={start_date}:{end_date}"
            ...
                response = requests.get(url)
                response.raise_for_status()
```

值得注意的是 API 提供的数据是按时间降序的，因此我额外编写了一个 $reverse(raw)$ 函数进行数据处理，把所有的数据存储在一个字典 data1 里，便于后续调用。

2.2 处理数据并绘图

这一部分对应 class 类中的 $plot(self, dict, countries, start\_year, end\_year)$ 函数，由于我们需要画多个折线图，且每个折线图里还有多个折线并且需要标明标签，因此我们直接考虑用国家的字符串 List 类与字典 data1 进行匹配，然后用循环绘制每一条折线，这边为了简便性，我们考虑直接把三张折线图保存在一张图片中，这样在 GUI 部分调取时会方便很多，即如下所示

[1]WorldBank:https://www.worldbank.org/en/home
[2]WorldBank API:https://documents.worldbank.org/en/publication/documents-reports/api
[3]Base API:https://api.worldbank.org/v2/country

```
fig, axs = plt.subplots(3, 1, figsize=(10, 30))
...
if not os.path.exists('image'):
        os.makedirs('image')
    axs[0].figure.savefig(f"image/out.png")
```

2.3 GUI 可视化和使用

这一部分对应 ___*init*___ 的主题部分，主要依托于内置的 tkinter 库进行实现，我在完成大作业时的主要困难基本全部集中在可视化部分，他们包括

- 窗口尺寸太小，导致输入输出排版异常混乱
- 在图片输出时，窗口尺寸往往会进行伸缩，且很容易出现无响应死机情况
- GUI 界面添加了 Logo 图像，同时还有输入和输出的很多控件，如何保证比较美观合理的布局设计和图片尺寸

这里我额外使用了 Pillow 库中 ImageTk 模块的 PhotoImage 类图片转换为适用于 Tkinter 的图像格式。结合 Image 类中的 resize() 方法保证合适尺寸，然后分配给 Tkinter 中的 Label 控件，使用 pack() 方法将其添加到界面中；另一方面，如果存在着页面缩放问题，可以考虑使用 Canvas 控件代替 Label 控件；对于界面大小的问题，通过查找资料，我在刚开始就固定页面大小，即把页面调整为

```
master.geometry("1200x900+100+100")
```

经过相当一段长时间的处理调整，我最终得到的界面大致如1所示。

Figure 1: GUI 界面

2.4 程序的鲁棒性和可扩展性考虑

在设计中我也充分考虑了输入的鲁棒性和可扩展性。同时为了程序的可读性，我也花了不少时间编写了 Readme.md 文档，尽我所能给我详细说明。首先在程序设计的时候我便设计好先把抓取额数据整理成一个比较“干净”的 dict 类，因此无论是折线图可视化的需求，还是其他的数据分析，都可以直接调用 dict 类进行分析，编写子类也很简单。第二点，由于一个国家的经济指标有很多，如果希望修改或者获取更多的经济指标，只需要在 utils.py 文件 99 行的 *datatype* 里添加对应的类或添加 *Args* 参数，即可轻松获取不同国家的经济数据并得到对比图像。

如果想获取其他数据，我们也可以直接修改 URL 的地址并且直接替换 LOGO，不同 LOGO 图片的尺寸不同，直接调用很有可能让界面变得混乱不堪，考虑到这一点，我们为 LOGO 尺寸设定了一个等比例的缩放，即

```
    logo_width, logo_height = self.logo_img.size
        logo_ratio = logo_width / logo_height
        max_logo_width = 400
    if logo_width > max_logo_width: #control the size and resize it to fit within the frame
            logo_width = max_logo_width
            logo_height = int(logo_width / logo_ratio)
```

可视化程序的鲁棒性也是很重要的一点，我们不希望一个程序输入后变得无响应死机，即使无有效输出也应即时报错。在程序中，若如果输入的国家或时间段有误，或者数据缺失，会给出错误提示，并直接在 GUI 输出“输入有误”；若在读取图像时没有找到文件，我们也会在界面输出错误。其中部分检验代码如下。

```
    except requests.exceptions.RequestException as e: #robustness
                    print(f"Error occurred while retrieving {key} data for {country}: {e}")
                    country_data.append([])
    except (ValueError, KeyError) as e:
                    print(f"Error occurred while processing {key} data for {country}: {e}")
                    country_data.append([])
    ...
    ...
    else: #robustness
            self.img_label.configure(text="输入有误，无法生成图像！")
```

3 实验收获与总结

作为一个 python 初学者，在完成全部代码跑通后，我深感自己代码质量和注释的不规范，于是学习了 Google 代码规范[4]，同时借助 Chatgpt[5]试着重构比较高质量的代码，例如我学会了在编写函数时先行指明输出类型，在检查输入和调用时使用 try 来及时报错等等。

```
def get_data(self,start_date: int, end_date: int, countries: List[str]) \
-> Dict[str, List[List[Tuple[int, float]]]]:
```

虽然本次实验耗费了我相当长时间的精力，学习了解了很多陌生的库和方法，但当所有知识拼接成功、代码跑通最终成为一个完整项目的时候，我知道在这门课上我学会了很多。最后也衷心感谢老师精彩的讲授和助教耐心的答疑。

[4] https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/
[5] https://openai.com/blog/chatgpt