实验报告

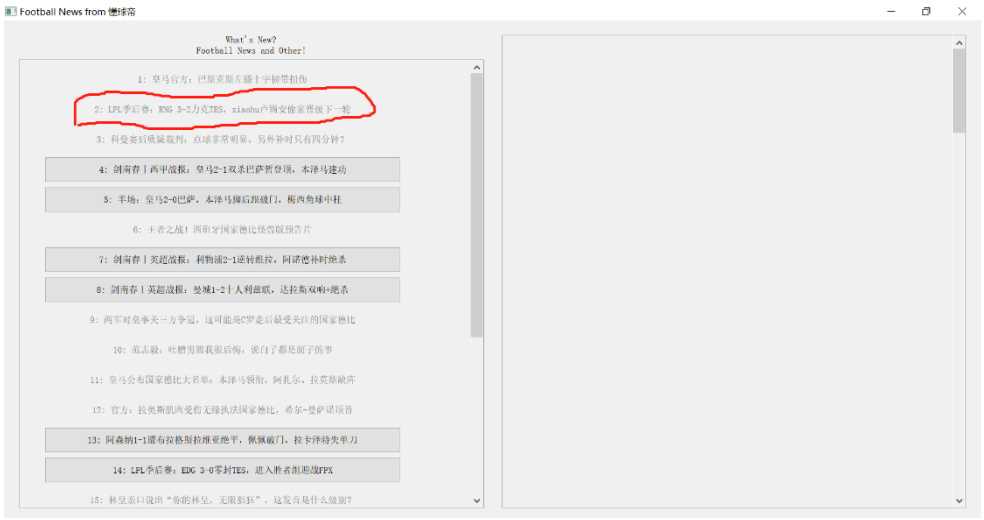
蔡方煦 PB18020500

1. 任务说明：

写爬虫从网站懂球帝的“热门”页面（网址 https://m.dongqiudi.com/home/104）爬取最近的热门新闻，使用 PyQt 将得到的新闻展示在自制窗口上，入下图所示：

窗口左侧为新闻展示，比赛战报为可被点击的按钮，点击后会从比赛战报所在网址上爬取关键时刻的比赛解说和比赛锦集 gif，展示在右边的窗口。（图中显示的是曼城 1-2 利兹联的战报，网址 https://m.dongqiudi.com/article/1941567.html）

非足球比赛的战报也可被点击，但是相关内容不会显示在右侧窗口，而且点击后按钮会变为不可点击模式，如下图所示：



2. 实验细节：

本代码中没有使用网站提供的 API，而是自己写爬虫对网页进行分析。

本代码中设计了一个类 FootballNews，其中包括的方法可以实现上一节所述的功能。方法__init__(self)初始化类，爬取网站 https://m.dongqiudi.com/home/104 的 html 源代码码，并从中提取新闻，然后再调用方法 FindGameReport(self)从新闻中提取比赛战报。

方法 GetGameDetails(self, game_report_url)从网址 game_report_url 爬取战报中关键时刻的锦集 gif 和解说（网页中下图所示的部分）。此方法被调用时，会首先判断参数中的网址

是否真的是足球比赛战报的网址，如果不是，则函数立刻返回，返回值为-1。解说（字符串）直接按顺序存在列表 self.game_report_content 中，gif 下载到当前目录的一个文件夹中，在列表 self.game_report_content 的相应元素写入字符串'gif'。

方法 MakeWindow(self)用于构建图形化窗口的基本框架。

方法 ShowNewsList(self)用于窗口左侧内容的构建，左侧的主体是一个可滚动窗口，其中放了一列按钮，每个按钮上的文字为一则新闻标题，如果新闻是战报，则按钮可被点击，否则按钮不可被点击。按钮被点击时调用方法 ButtonClicked(self)。

方法 ButtonClicked(self)首先通过方法 sender()得到发出信号的按钮，然后在通过这个按钮预先设好的名字(序号)判断这个按钮对应的是第几条新闻，从而得到新闻（战报）的网址，调用方法 GetGameDetails()。如果返回值是-1，意味着按钮对应的并不是足球战报，于是此按钮被禁用，同时清空右侧展示内容的窗口。如果按钮确实对饮足球战报，则调用方法 ShowGameReport()。

方法 ShowGameReport(self)将预先爬取的数据展示在右侧的可滚动窗口中。所有内容都通过 QLabel 展示，如果是 gif，需要在 QLabel 上添加 QMovie 类。

遇到的问题与（可能的）解决方案：

1. 网页 https://m.dongqiudi.com/home/104 是动态加载的，浏览器中新闻加载的条数（以及 html 源代码的长度）是随浏览进度增加而增加的，requests 只能静态爬取网页 html 代码，只能加载出 20 条左右的新闻。
   解决方法：暂时没有，估计得用其他的库。

2. 如何判断一条新闻是不是比赛战报，从新闻列表的 html 代码看来，并没有有区别的特征。如果要打开一个一个链接判断太耗时了。
   解决方法：直接从新闻的标题看，标题中有比分 XX-XX 的就认为是比赛战报。当然，有时候这样会误判，由于懂球帝是不是会有一些足球无关新闻，比如篮球、电竞，所以爬下来的战报不一定是足球比赛的。可能的进一步解决方法：判断标题中有没有出现足球俱乐部名、国家队名。最后，我放弃了在这一步对战报进行更细致的区分，而是在后面要进入链接查看关键事件时再做区分。

3. 爬取战报中的 gif 速度慢。
   解决方法：这受限于网速，没什么办法。但有一个可能的优化方法：把不同战报存

下的 gif 放到不同的临时文件夹中，下一次查看时就不必再到网上爬了。（目前只设了一个临时文件夹，查看新的战报时原来的 gif 可能会被覆盖）

4. 有些战报中会插入广告，要把这些广告和 gif 区分
   解决方法：多加一条 if 语句。

5. 如何判断窗口中是哪个按钮被点击了
   解决方法：给每个按钮用序号命名（名字不是按钮显示的文本），使用 sender()方法返回被点击的按钮实例，再查询这个实例的名字就好了。

6. 所有窗口中播放的 gif 比下载下来的 gif 速度明显更慢
   解决方法：没办法，手动设置 gif 播放速度变为 1.25 倍数，这估计与实际速度还是不符，但人眼看区别不大了。

7. 窗口中的 gif 放大后比例失调，即使用了 Qt.KeepAspectRatio 也没有用，实验也证实不是 QLabel 的尺寸造成的限制。
   解决方法：手动设置所有 gif 到黄金分割比，这是网站上大部分战报的 gif 的长宽比，但也有部分战报不是这个比例。

8. 我使用的是 PyQt5 的库，但发现网上很多参考例子使用的是 C++或者 PyQt4 的库，所以部分网上的解决方案并不适用。

3. 实验总结：

在本次实验中我从 0 开始学习并尝试了编写网络爬虫和使用 PyQt5 构建窗口。除去学习时间，代码编写中最花时间的部分是：试图用 requests 去动态加载网页，最后失败（网上实际上有例子，不过看上去比较复杂）；用 PyQt 进行各种排版以及控件调整。

经过此次实验，我最大的收获是：学习了 PyQt 代码进行布局的核心思路，初步学习了网络爬虫的相关知识。