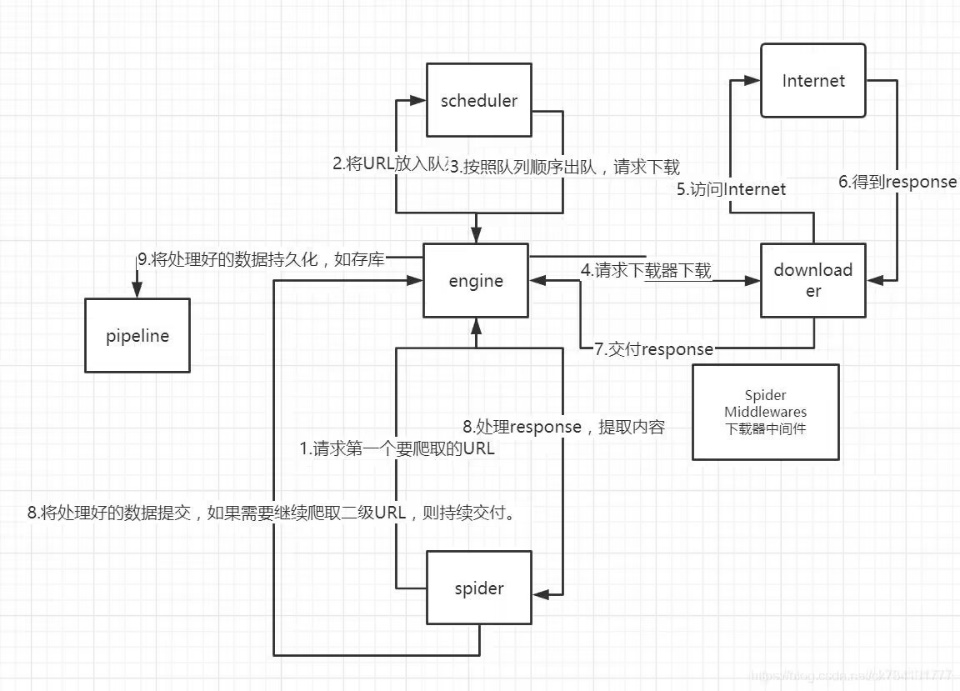
## 一、项目分析

我们本次数据库课设题目为基于AI的新闻热点聚合及可视化分析，经分析后，我们可知本次项目的起点就是新闻的爬取。鉴于本学期正好选修python程序设计，且该语言支持不少方式的爬虫技术，我们选用python对新闻内容进行爬取。

## 爬虫技术适应

### Scrapy框架

Scrapy框架是python开发当中的一个快速、高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面提取结构化的数据。其由五大组件组成，分别为调度器(Scheduler)、下载器(Downloader)、爬虫（Spider）和实体管道(Item Pipeline)、Scrapy引擎(Scrapy Engine)，运用Scrapy框架时，程序发生的框架图如下所示。



图表 1整体框架图

流程如下：

1）爬虫中起始的 URL 构造成 Requests 对象 爬虫中间件 引擎 调度器；2）调度器把 Requests 引擎 下载中间件 下载器；

3）下载器发送请求，获取 Responses 响应 下载中间件 引擎 爬虫中间件 爬虫；

4）爬虫提取 URL 地址，组装成 Requests 对象 爬虫中间件 引擎 调度器，重复步骤2；

5）爬虫提取数据 引擎 管道处理和保存数据；

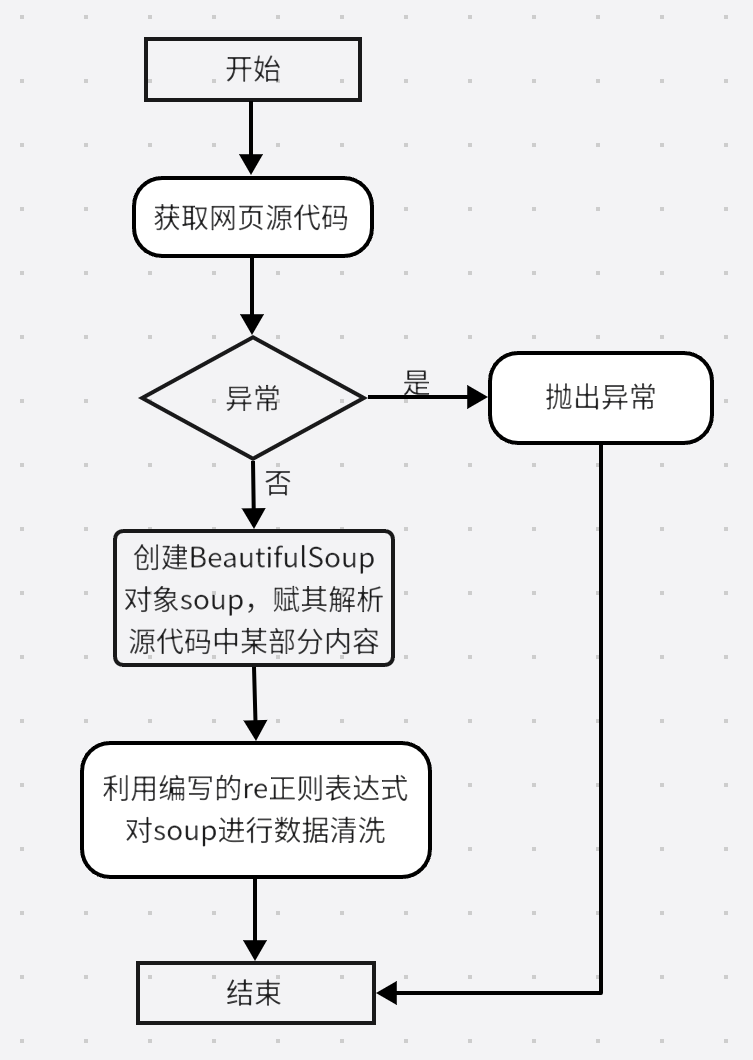
### 2、bs4+re+urllib

bs4，即BeautifulSoup，是python中用于爬取网页数据的一个库。Beautiful Soup提供一些简单的、python式的函数用来处理导航、搜索、修改分析树等功能。它是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。Beautiful Soup自动将输入文档转换为Unicode编码，输出文档转换为utf-8编码。不需要考虑编码方式，除非文档没有指定一个编码方式，这时，Beautiful Soup就不能自动识别编码方式了。

re库是python用于正则表达式的第三方库。正则表达式是用来简介表达一组字符的表达式，由字符串和操作符构成。

urllib是python中一个用于操作URL的库，其主要有请求模块、异常处理模块等。

总体思想即为借助urllib中的request模块对给定url发起请求，获取到网页源代码，然后利用bs4的BeautifulSoup对象解析源代码内容，获取某部分源代码，对获取到的这一部分源码再使用re正则表达式解析，即数据清洗，则获得最终想要的内容。流程图如下：



图表 2 bs4+re+urllib流程图

## 借助爬虫技术对某新闻网站进行爬取

1. 方式选择

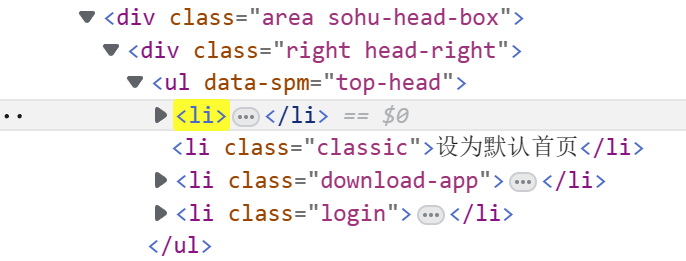
由于Scrapy框架技术较繁琐，且需要利用浏览器驱动器，配置繁琐且不安全，我们暂不考虑使用该框架。其次，本次爬取内容并不是海量爬取，利用bs4+re+urllib就可以满足较快且完整地爬取。综上，我们尝试利用后者对搜狐新闻、观察者网的主页进行新闻爬取。

1. 搜狐新闻网站爬取过程

我们打开搜狐新闻网站，在浏览器中检查该页面，并利用元素选择找到我们需要爬取内容的源代码部分，如下所示：



我们由此结合javaweb的知识可知，该新闻网址在<li>标签中，为了检验其是否为偶然，对其他新闻也进行了检查，发现所有新闻网址都处在<li>标签当中，但这还远远不足，我们搜索所有<li>标签，如下：



可知并非所有<li>标签中的内容都是我们想要的，但我们想要的内容却又都在该标签当中，该如何？当然是继续缩小解析范围了。除却标签外，class属性也是一个筛选的重点，因为细心观察后发现，绝大多数新闻内容都在list16这个类中，如下：



因此我们又多了一个筛选条件。我们创建一个item对象，将html中所有类为list16的内容都赋给它，但这不够，我们再创建一个item1对象，将item中所有li标签的内容都赋给它，这样，item1就包含了我们所有想要的信息了！

接下来我们对item1中的内容进行进一步提取：观察后可知，所有li标签中的内容包含了新闻的链接以及标题，此时我们就需要用到re正则表达式来对其进行提取了，首先对照多条li标签内容得出规律，我们想要的新闻链接在li标签中href=“…”的双引号当中，我们可以编写该部分的正则表达式为findlink = re.compile(' href="(.\*?)"')，很遗憾，我们不能继续获取到标题信息了，因为标题所在位置并不如链接一般有规律，无法编写其正则表达式，但我们可以进入该篇新闻的网页，跟上述一样获取到它的网页源代码并提取有用信息，我们一试。

我们进入某新闻页面后，看到该新闻标题所处的位置在h1标签，所属text-title类中，再看其他新闻， 也是同样的在h1标签当中，但是标题前后的属性有些参差不齐，要编写正则有些难度，无法统一，那就无法一次性解析出所有标题了，必须找出它们的共同点。可以看到，h1标签中，除了标题外，其余内容都在<>当中，那么我们可以利用re正则表达式将h1标签内容中所有的<>以及其中的内容去掉，剩下的就是标题内容了。Re.sub（）方法就提供了这样了一个功能，将<.\*?>替换为空字符就好了，这样，就完美的提取出了标题内容。

继续获取正文内容。如下是正文部分对应的源代码：



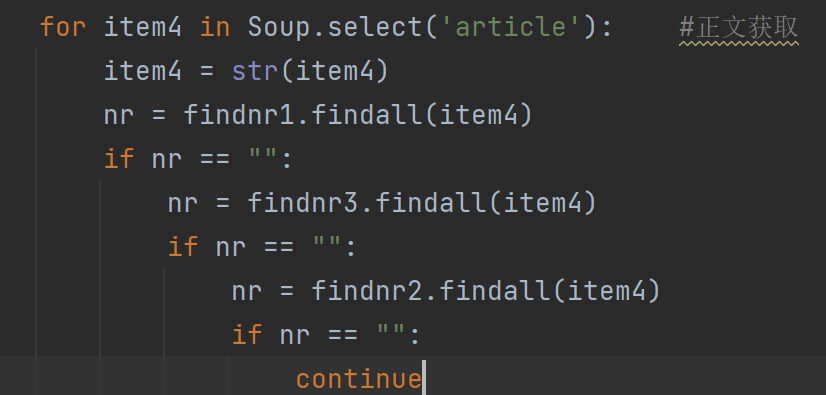
我们看出，正文内容全都在p标签中，对照其他篇，发现有所不同，不少文章的正文都在<p class="ql-align-justify">…</p>和span标签当中，我们可以编写三个正则表达式，分别为

findnr1 = re.compile('<p>(.\*?)</p>')

findnr2 = re.compile('<span>(.\*?)</span>')

findnr3 = re.compile('<p class="ql-align-justify">(.\*?)</p>')

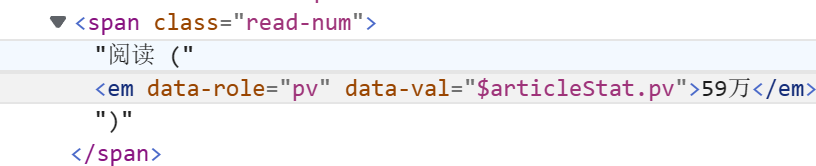
这里有个细节，就是span标签有部分是原标题内容，所以我们先将p标签的所有内容放入一个对象中，判断其是否为空，若为空则将所有<p class="ql-align-justify">标签的内容放入该对象，此时若仍为空，才将span标签放入，第二步与第三步不能颠倒顺序，否则可能会只爬取到原标题的内容，而非正文。这里还有个细节，那就是有少量新闻仅仅是一幅图或者一个视频，这类新闻我们选择不怕，因此我们在最后还需要对该对象进行非空判断，若为空，则continue执行下一个for语句，具体如下：



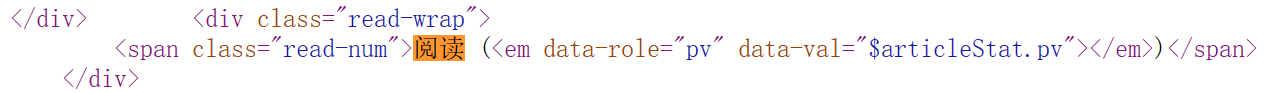
当然，在正文中还有些多余的字符出现，我们需要进一步清洗，利用python当中字符串的replace函数将这些字符一一替换为空值即可。至此，我们的数据基本获取完成了。

由于本次我们的项目还需要对新闻热度进行一个分析，那么这对于爬虫部分就有了新的要求，就是爬取的内容当中，需要有一个值来代表某篇新闻相对的热度，有以下数据可以参考，分别是浏览量、点赞量、评论数，显而易见，浏览量就是最好的体现，但问题是不少网站都不提供浏览量的数据，只有寥寥几个，我们找出了两个网站，分别是搜狐视频以及观察者网。接下来，我们先继续对搜狐视频的浏览量进行爬取。

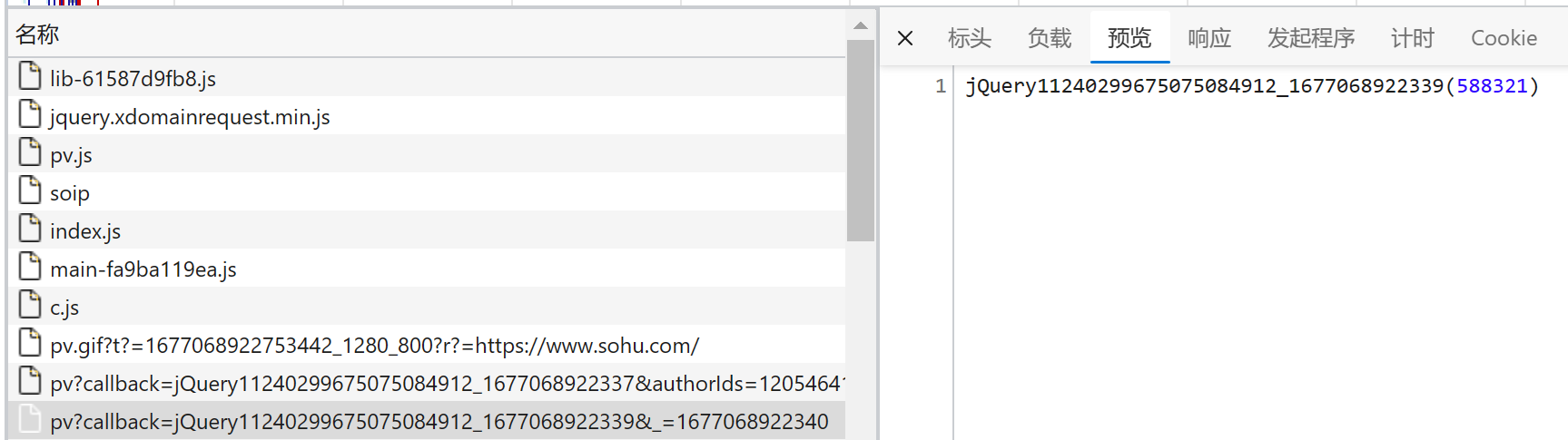
我们找到浏览量所在的源代码处，如下：



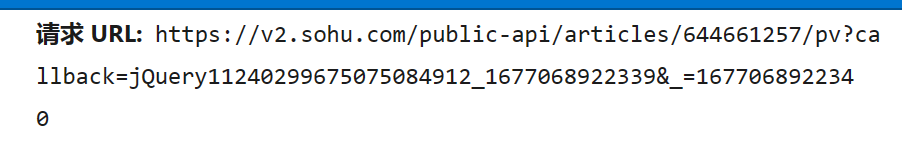
我们当然是依葫芦画瓢地按照上面的方法对“59万”进行爬取，但我们进行输出的时候发现其为空，问题出现了，查看网页源代码，找到对应位置地部分，结果出乎意料：此处为空



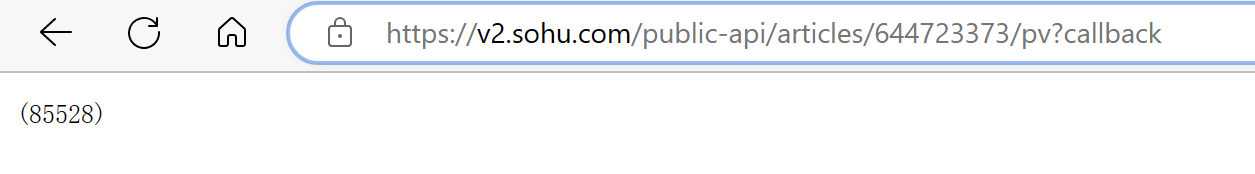
我们猜想，这是一个动态加载的数据，在网页源代码中无法直接获取，这时候就需要进行一个抓包操作了。在对页面进行刷新后，查看其网络，对那里的包进行查找，这里我们观察到原页面的data-role为pv，我们寻找出有pv字段的包进行一一预览，果然找到了它！



多篇新闻对比后，确定它就是该新闻的浏览量数据，但是如何抓取下来呢？这是一个js包，按理说我们应该可以得到其内容，但是方式不知道是怎样的，只能继续想办法了。这时候，我们点击标头，可以看到这里有该js包的请求url

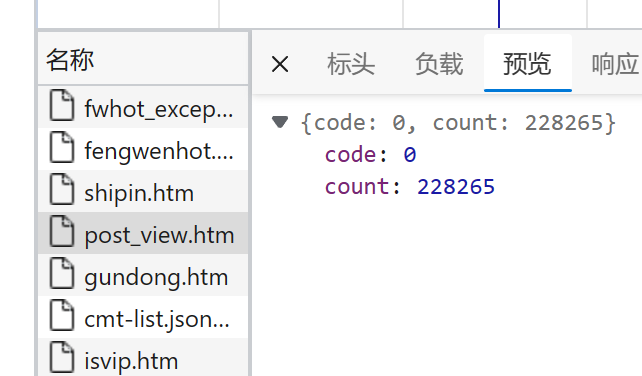


我们也许可以从这里下手，解析出这个url的构成规则，从而得到每篇新闻单独对应的浏览量的js包网址。查看其他多篇新闻的该包url发现，artivles/后的这一串数字以及=jQuery及其后半部分的内容是每篇新闻都不一样的，且观察该篇新闻的链接后发现那一串数字在该链接当中，我们可以对前面所得的链接进行提取，编写re正则表达式为findrn = re.compile('/a/(.\*?)\_')，提取该部分数字，然后用字符串的方法连接起来得到一个新的url，此处去掉了jQuery后部分的内容，因为这里显然是一个传值的过程，将值传给jQuery的js包中，我们直接删掉得到数据所在url，再进行访问，可以看到：



这里我们直接可以看到网页信息就是浏览量，可以直接获取。至此，搜狐新闻所需的数据就全部爬取完成了，下面对观察者网的爬取进行简要阐述。

方法相同，在对观察者网进行爬取的时候，可以借助上面的框架进行修改即可，但同样到了最难的浏览量爬取时就遇到了大问题，方法一样是在网络中查找有可能出现浏览量数据的包，这次很轻松地就在page\_view.htm包中找到了，如下



但观察标头后就发现，这个页面的请求url是一个固定的网址，是通过请求链由一个js包传递的数据，这就无法直接得到直观的数据展示的页面url了，这就难倒一个入门选手了。在一番挣扎后，我们选择放弃爬取浏览量作为热度数据。

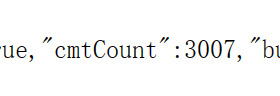
我们还未解决另一个新闻网站的热度数据爬取，因为展示浏览量的网站太少，这里我们退而求其次，选择爬取新闻评论数作为热度分析数据。评论数在每个网站都是可以展示的数据，因此我们放弃了观察者网，选择继续爬取另一个门户网站：网易新闻。接下来介绍爬取网易新闻评论数的过程，其余部分与上述大同小异，不再赘述。

网易新闻的评论数在页面中的展示如下：



我们直接检查该部分代码：



可以知道，这里显示的数据一定是动态加载的，验证后果不其然，在网页源代码中并未显示该数据，但这里给了一个url地址，我们尝试转到该页面：这是该篇新闻的评论详情页，显示了热门评论以及总参与数，我们直接转到该页面的源代码，查找显示该数据的部分，但如预想般的一样，并不展示。我们尝试在网络中查找，但遍历一遍后，未发现明显的展示，只能重新回到该源代码的部分。我们知道，这样动态加载的数据有可能是函数传值，在各个包中查找显然是不现实的，我们只可能在源代码处碰碰运气，观察是否在其中的某个函数当中，一番查找后，幸运地发现了其所在处：，位于function（）函数中。这是与html分离的一部分内容，我们无法使用bs4进行数据解析，只能选择用re正则提取评论数。编写其为：

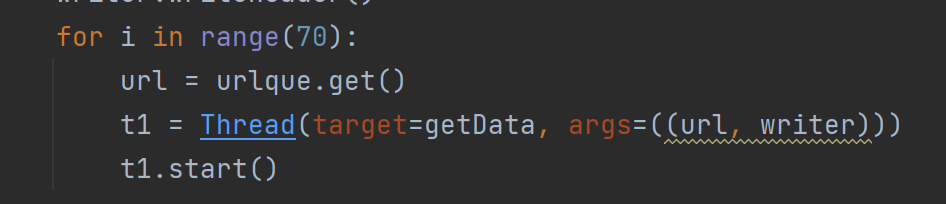


在这之前有个小步骤，就是提取评论详情页的url，与上述爬取搜狐新闻浏览量时提取浏览量网址时步骤相同，不再赘述。

1. 代码优化及补充

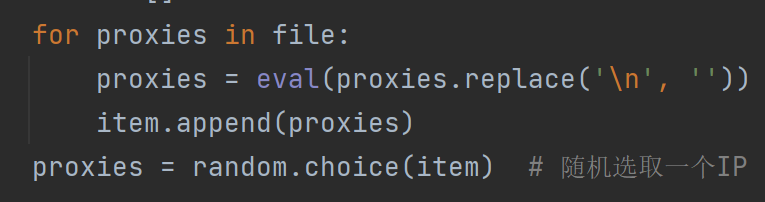
上述已经实现对所需数据的爬取，但作为程序编写者，还必须考虑到代码的效率。对其添加time库计时后发现，爬取百余条新闻用时接近一分钟，这是丑陋的，对此，必须改进代码。

首先考虑整体架构，由于没有采用Scrapy框架，而采用了简单的bs4+re正则的方式，在本身就有了差距，但我们还是相出了一个结局方法：多线程爬取。我们引入threadi模块中的Thread类，可以借此开启多个进程，实现并发爬取，可以大大提高爬取效率。但这里需要对代码进行整改一番，首先是爬取步骤，在之前，我们是先由新闻网站得到某篇新闻的网址，爬取该网址后随即便访问该网址，对其中的标题、正文、发布时间进行爬取，最后访问浏览量所在网址，由此我们发现，这里有嵌套的现象，那么其时间效率就是小于等于n^2，我们需要将其拆开，解决方法是先爬取所有的url地址，存入一个列表中，再从该列表一一取出url进行访问，然后爬取，这样，时间效率就有了跨度性的提升了。这里，我们还引入了队列，可以将url一一扔进队列，当进行其他数据爬取时，将队列中的url一一抛出即可，且队列先进先出的特性符合先爬到的url先访问。开启70个线程，对队列中的url地址一一访问爬取，如下：

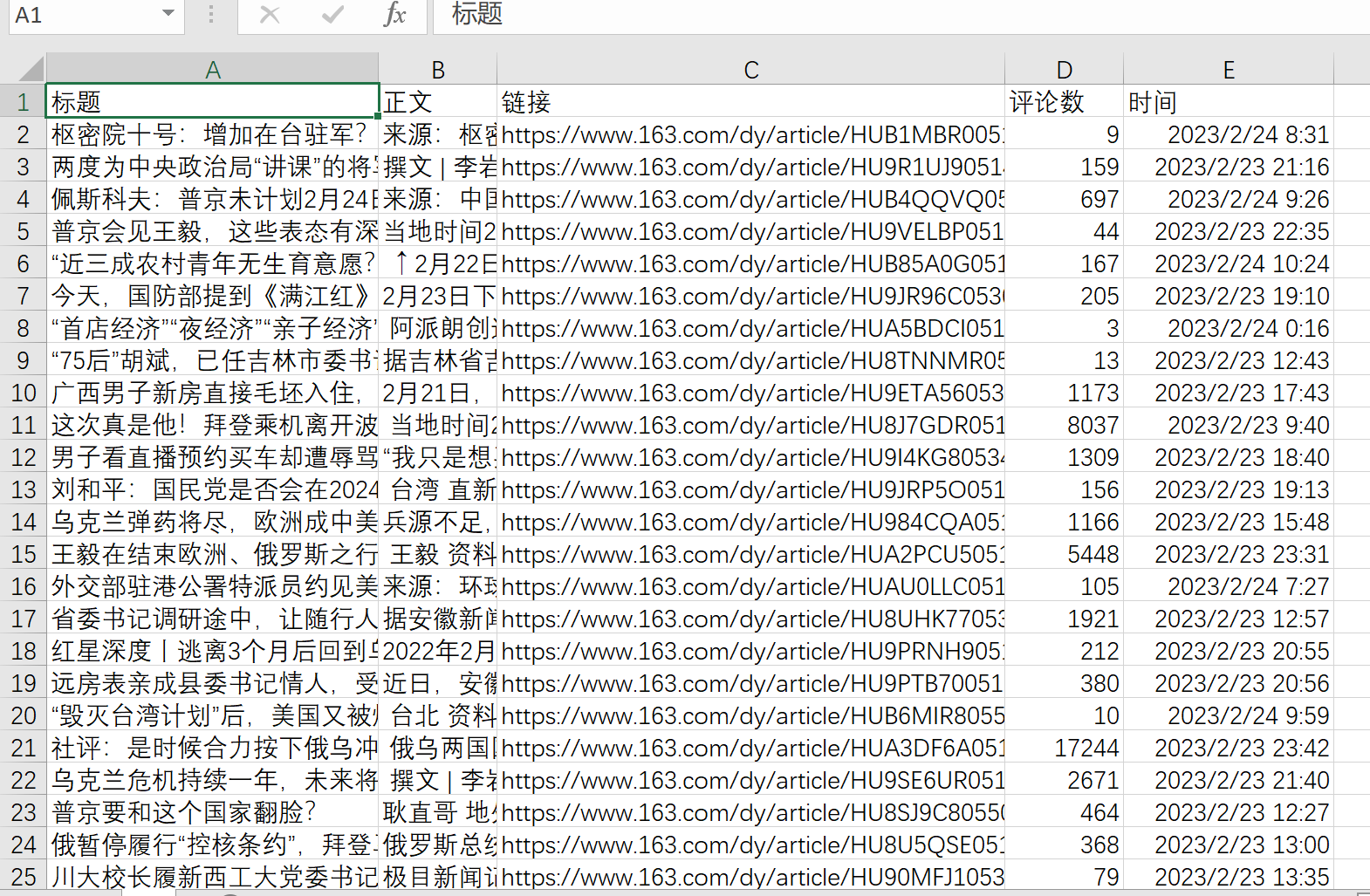


最后，爬取符合的新闻条数大约为50，用时在1~2s，与之前的效率相比有质的提升。

至此，我们的任务还未结束，因为爬虫工作需要面临各个网站的反爬虫机制，我们在此前即便伪装了请求头也无济于事，毕竟我们在调试程序的时候需要不断访问，自然也就被服务器检测到这一不合常理的操作，经常被封。为了解决这一问题，我们尝试使用代理IP池伪装自己，实现多次访问。我们在网上找到了一个提供免费IP的网站，那我们就利用爬虫技术将其爬取下来，存入一个IP.txt文件中，在上述代码中requests.get（）中添加proxies参数为IP，这个IP随机选取IP.txt文件中的IP即可。如下：



最后，展示爬取到的新闻内容：



至此，爬虫工作完成。