本爬虫使用 selenium 做为报文的抓取工具，使用 BeautifulSoup 做为报文的解析工具，并未使用进阶的技巧，只是控制爬取行为尽量与正常用户相似，以此逃避反爬检测。

爬取流程：

1. 首先执行 get_cookie()，该函数执行的操作是打开 jd 的登陆页面，需要手动登陆，登陆成功后会自动把 cookie 保存在本地的 json 文件中，通过判断页面的 title 是否为'京东(JD.COM)-正品低价、品质保障、配送及时、轻松购物！'来判断是否登陆成功，之后将执行下一步。保存 cookie 是为了使得之后在访问 jd 时，只要导入 cookie 就无需在次登陆了，判断 title 时用到了 selenium 的 WebDriverWait 和 expected_conditions 两个组件，前一个用来等待某个条件发生，发生后才会继续执行，若规定时间内未发生则会抛出异常。后一个是条件判断组件，用于判断某个条件是否发生，WebDriverWait 和 expected_conditions 的组合在 selenium 中比较常用到。
2. 之后的操作由三个进程相互配合完成，通过信号量控制进程间的推进顺序和文件的互斥访问，第一个进程通过商品名'蓝牙键盘'获取具体商品列表的报文，第二个进程通过前一进程获取的列表报文，提取具体商品的链接，通过链接访问商品获取商品页面的报文，并通过模拟点击页面的操作，进行评论页的翻页操作，获取更多的评论。第三个进程通过解析前一进程获取的报文，从而获取用户的评论。
   a. 第一个进程
      首先通过 xpath 定位商品搜索框，键入'蓝牙键盘'并按下搜索键，之后通过随机数执行随机数量的下拉页面和等待操作，之后获取报文并键入右方向键执行翻页操作，重复以上操作直到获取到预定数量的页面。模拟按键使用的是 selenium 的 ActionChains 模块，该模块需先初始化一组操作，之后执行.perform()则可将预先初始化的操作按顺序执行，是 selenium 常用的模拟按键工具。
   b. 第二个进程
      将第一个进程保存的报文通过 BeautifulSoup 解析，根据字符串'strong'、'a'和属性"href"定位具体商品的链接，之后访问该链接，不断执行翻页和保存报文的操作，直到评论均爬取完毕。翻页操作是通过点击所有具有下一页页数的字符串和属性"rel"的单元，比如下一页为第四页则需要定位带有内容'4'和属性"rel"的单元。
   c. 第三个进程
      该进程是将上一进程保存的报文，通过 BeautifulSoup 解析，根据标签'p'和 css 类'comment-con'定位评论内容，解析出评论内容并利用 json 保存在本地。
3. 结果
   爬取结果如图 2 所示，爬取过程比较慢，如果想加快速度可以使用 IP 池多开几个并行爬取。另外需要注意的是，尊重别人的隐私，爬取的数据仅用于学习不要外泄，用完就删掉，涉及到 ID 的数据不要爬取。本次实验爬取到的数据在保存后就删除了。

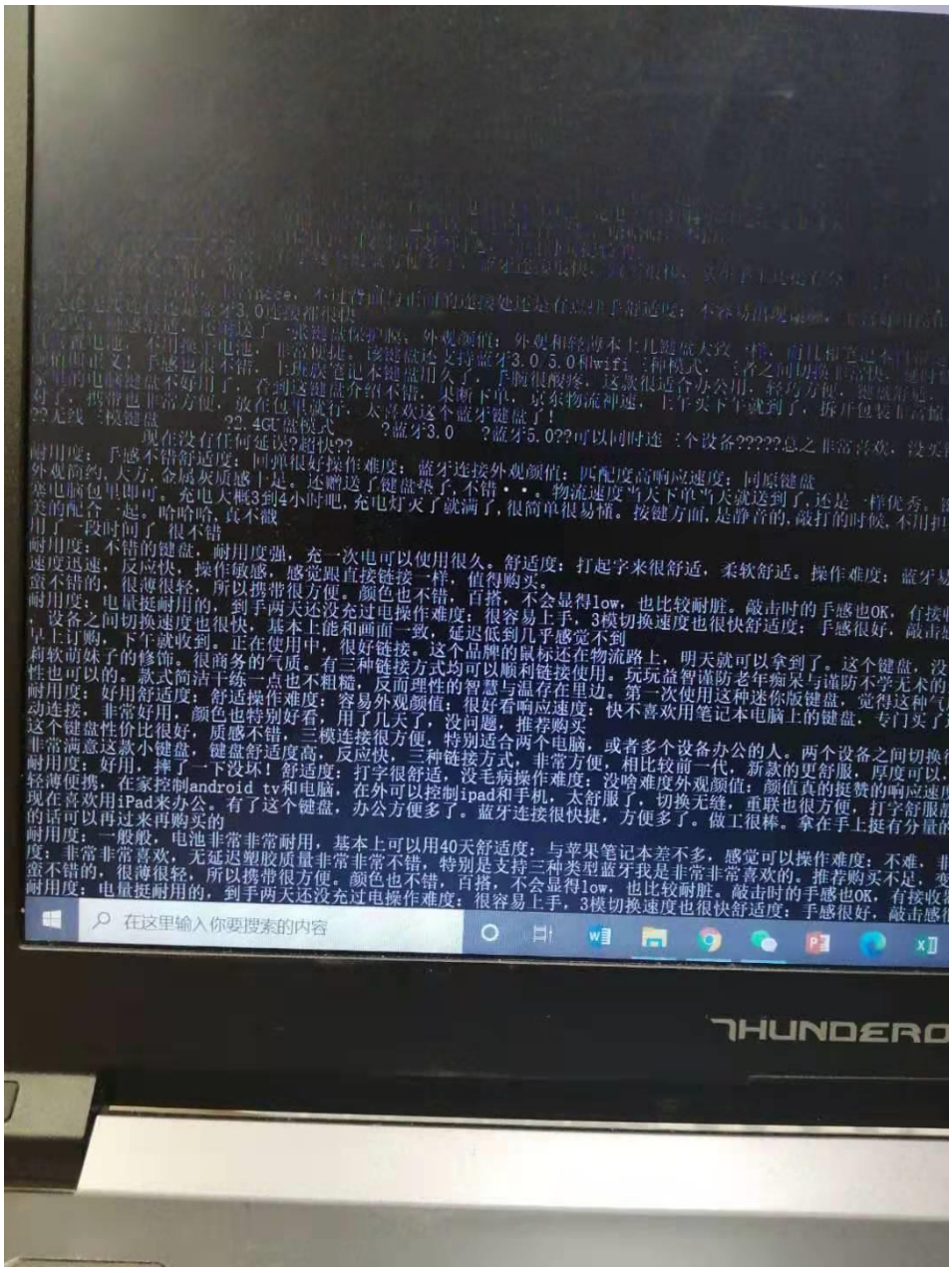


图 2