**技术文档**

**一．淘宝商家信息的爬取**

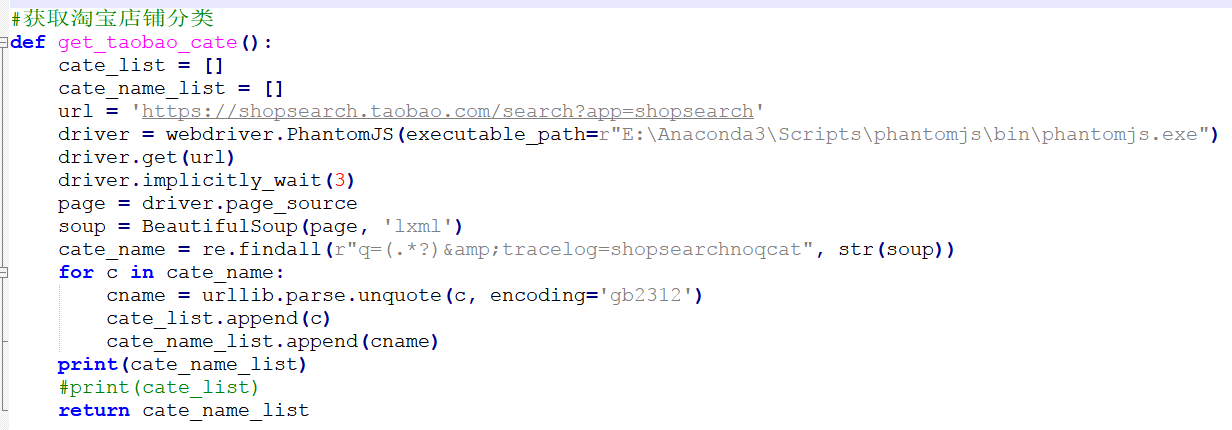
# 利用的是selenium中的webdriver.phantomjs获取淘宝店铺分类页面信息。通过BeautifulSoup 对页面信息进行分析，获取到具体的商铺信息。

**PhantomJS：** 基础的爬虫只能爬取单纯的html代码，但是有很多页面是js渲染的，html源码里面神马都没有。这是就需要借助于像浏览器一样渲染js页面的工具了。PhantomJS是一个无界面的,可脚本编程的WebKit浏览器引擎。它原生支持多种web 标准：DOM 操作，CSS选择器，JSON，Canvas 以及SVG。

**Beautiful Soup：**是一个非常强大的工具，爬虫利器。一个灵活又方便的网页解析库，处理高效，支持多种解析器。利用它就不用编写正则表达式也能方便的实现从 HTML 或 XML 网页中抓取信息。

**爬虫部分代码：**获取淘宝店铺分类后通过 soup = BeautifulSoup(page, 'lxml')解析获取到的网页信息，通过正则表达式获取商铺分类信息：

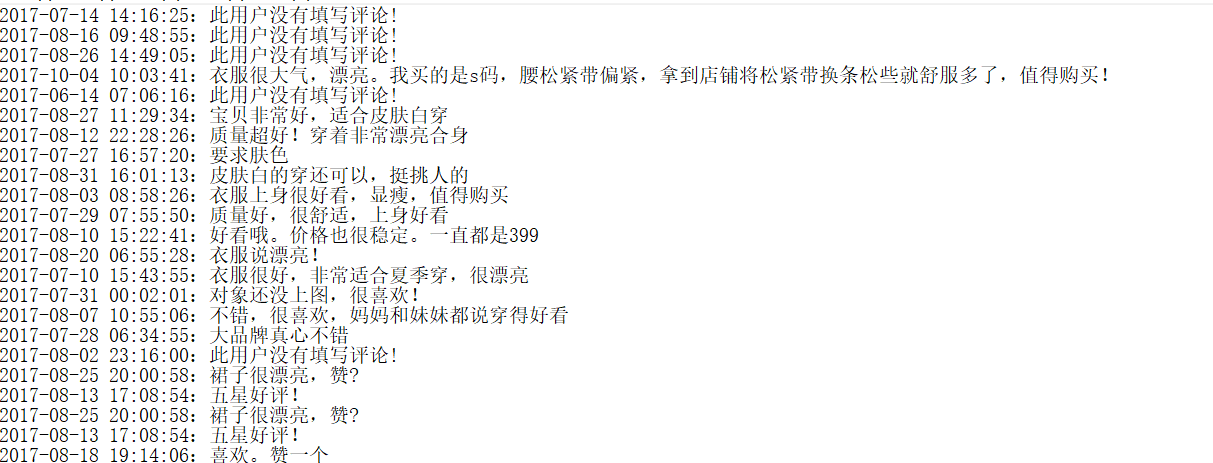
cate\_name = re.findall(r"q=(.\*?)&amp;tracelog=shopsearchnoqcat", str(soup))



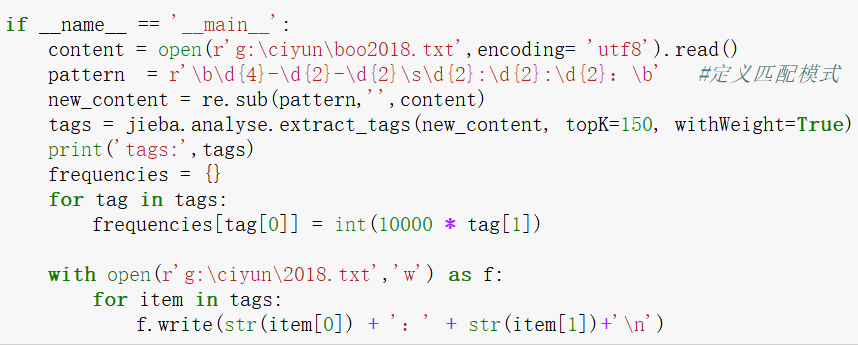
**爬取淘宝商铺信息：（由于数据量较多淘宝爬取不稳定所以这一部分的信息来源网络）但是下面的代码是可以成功爬取到商铺信息的。**

**二．数据的整理**

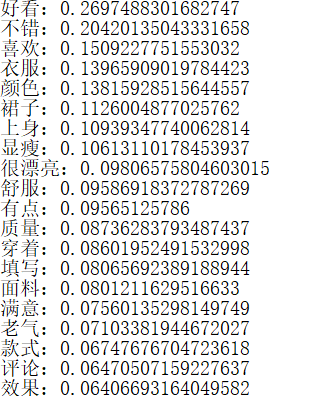
对爬取的数据进行整理，爬下的部分数据如图所示:



首先利用正则匹配，提取出评论的文字部分，接着利用结巴分词，对数据进行关键字提取，部分代码如下：



提取出如下关键字：



**三．网站的搭建**

我们的在搭建flask网站的同时，结合pyecharts与bootstrap做成了一个简易的可视化网站，flask后台发送数据，前端接收数据并用pyecharts进行展示。

**3.1．pyecharts**

pyecharts 是一个用于生成 Echarts 图表的类库，ECharts，缩写来自Enterprise Charts，是商业级数据图表，一个纯Javascript的图表库，是我们比较熟悉的一个简单的数据可视化工具。而pyecharts实际上就是 Echarts 与 Python 的对接，提供了直观，生动，可交互，可高度个性化定制的数据可视化图表。

pyecharts支持折线图（区域图）、柱状图（条状图）、散点图（气泡图）、K线图、饼图（环形图）、雷达图（填充雷达图）、和弦图、力导向布局图、地图、仪表盘、漏斗图、事件河流图等12类图表，同时提供标题，详情气泡、图例、值域、数据区域、时间轴、工具箱等7个可交互组件，支持多图表、组件的联动和混搭展现。

用户需要自定义：

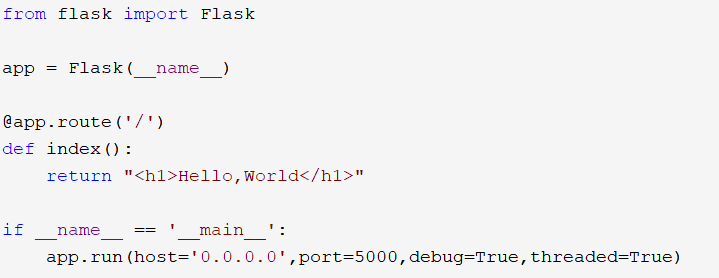
Grid 类：并行显示多张图   
Overlap 类：结合不同类型图表叠加画在同张图上   
Page 类：同一网页按顺序展示多图   
Timeline 类：提供时间线轮播多张图

**3.2 flask网站**

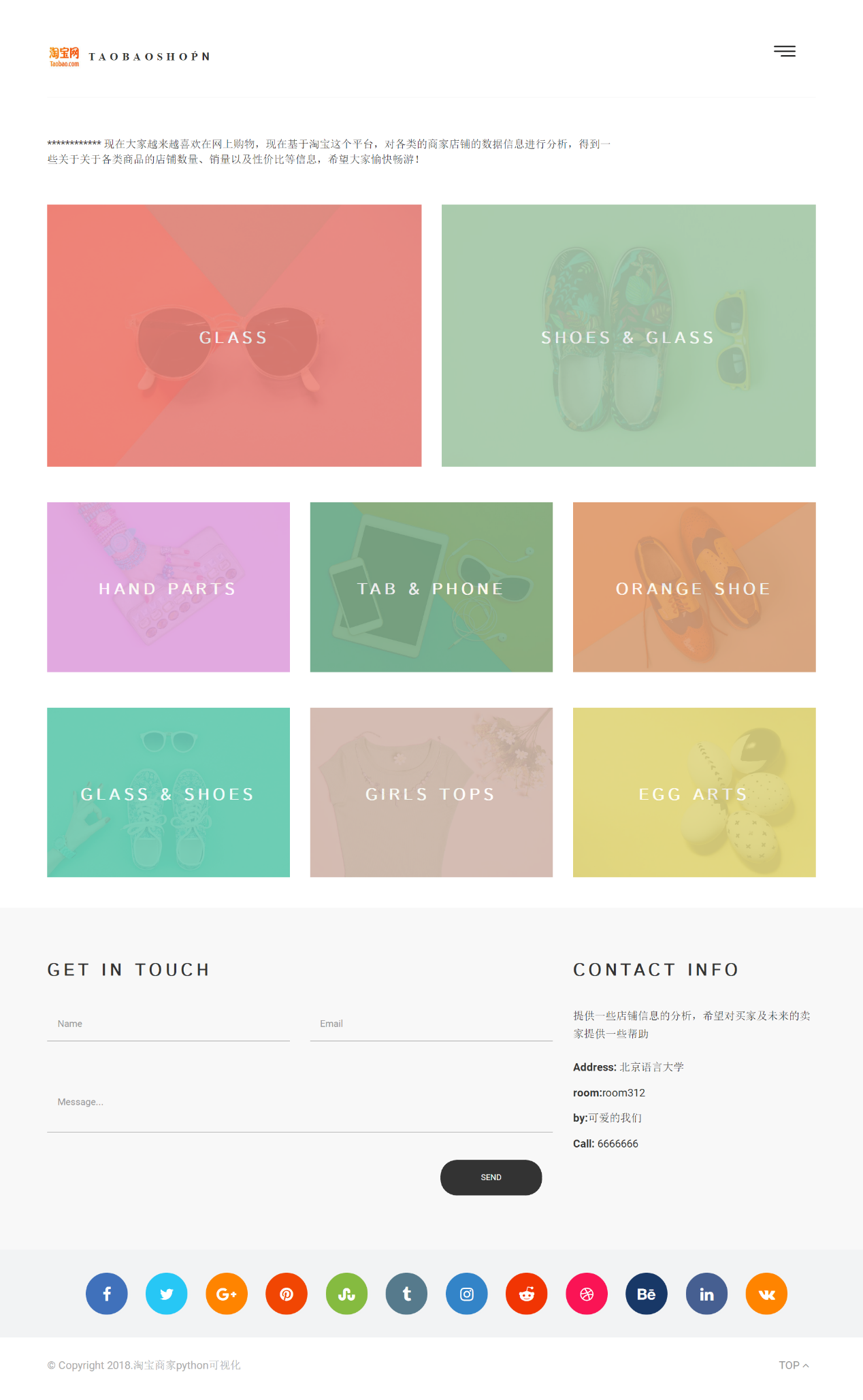
Flask是一个基于python的，微型web框架。之所以被称为微型是因为其核心非常简单，同时具有很强的扩展能力。它几乎不给使用者做任何技术决定。

默认情况下，Flask 不包含数据库抽象层、表单验证，或是其它任何已有多种库可以胜任的功能。然而，Flask 支持用extension来给应用添加这些功能。众多的扩展提供了数据库集成、表单验证、上传处理、各种各样的开放认证技术等功能。

　比如最简单的一个flask的web应用如下：



**3.3 bootstrap**

****

**3.4 bootstrap**

Bootstrap，是基于 HTML、CSS、JAVASCRIPT 的，它简洁灵活，使得 Web 开发更加快捷。它提供了全面、美观的文档。在这里能找到关于 HTML 元素、HTML 和 CSS 组件、jQuery 插件方面的所有详细文档。

**3.5 搭建步骤**

Step1. 新建一个 Flask 项目project，新建一个app 文件夹，在其下新建 templates 子文件夹。

Step2. 为在template文件夹下放入自己选择的网站模板boostrap文件。

Step3. 将css，js，images等静态文件移至static文件夹下。

Step4. 使用pyecharts进行可视化，将相应的文件放入到app/mycharts文件夹下

Step5. manage.py进行启动运行。

此时 project 目录下为：

├── manage.py

└── app

|\_\_\_\_\_\_\_\_\_all\_date\_files

|\_\_\_\_\_\_\_\_\_mycharts

|\_\_\_\_\_\_\_\_\_static

|\_\_\_\_\_\_\_\_\_templates

|\_\_\_\_\_\_\_\_\_data.py #获取数据文件

|\_\_\_\_\_\_\_\_\_views.py

|\_\_\_\_\_\_\_\_\_ \_init\_.py