Changjiang Securities

2019.04.08 - 2019.06.17

As an intern of Industry Research Department in Changjiang Securities, I am mainly responsible for data collection and supporting, our team's focus was on retails including shopping malls, beauty brands, baby cares, etc.To quickly attain the number of stores for large chain brands, a scrapper was written to automatically calculate from the Baidu Map, greatly improving our team's efficiency

In [1]:
```
import urllib
import requests
import pandas as pd

def Crawl(url):
    headers = {"Accept": "*/*",
          'Accept-Encoding': 'gzip, deflate, br',
          'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
          'Connection': 'keep-alive',
          'Host': 'map.baidu.com',
          'Referer: https': '//map.baidu.com/search/%E4%BC%98%E8%A1%A3%E5%BA%93/@13432005.56,3644785.89,13z?querytype=
s&c=224&wd=%E4%BC%98%E8%A1%A3%E5%BA%93&da_src=shareurl&on_gel=1&l=13&gr=1&b=(13401285.56,3629281.89;13462725.56,366028
9.89)&pn=0&device_ratio=2',
          'Sec-Fetch-Dest': 'empty',
          'Sec-Fetch-Mode': 'cors',
          'Sec-Fetch-Site': 'same-origin',
          'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
/83.0.4103.61 Safari/537.36'
          }
    r = requests.get(url, headers)
    return r.json()

def getData(url):
    currentInfo = []
    moreInfo = []
    data = Crawl(url)
    current_province = data['current_city']['up_province_name']
    current_city_info = data['content']
    other_city_info = data['more_city']
    for city in current_city_info:
        l = []
        city_name = city['view_name']
        city_num = city['num']
        l.append(city_name)
        l.append(city_name)
        l.append(city_num)
        currentInfo.append(l)
    for other_cities in other_city_info:
        province = other_cities['province']
        cities = other_cities['city']
        for city in cities:
            l = []
            other_city_name = city['name']
            other_city_num = city['num']
            l.append(province)
            l.append(other_city_name)
            l.append(other_city_num)
            moreInfo.append(l)
    return currentInfo + moreInfo

def sumData(url):
    Info = getData(url)
    Info_df = pd.DataFrame(Info)
    Info_sum = Info_df[2].sum()
    return Info_sum

def getUrl(target_name):
    target_store = urllib.parse.quote(target_name, encoding = 'utf-8', errors = 'replace')
    target_url = 'https://map.baidu.com/?newmap=1&reqflag=pcmap&biz=1&from=webmap&da_par=after_baidu&pcevaname=pc4.1&q
t=s&c=1&wd=' + target_store + '&da_src=pcmappg.map&on_gel=1&l=5&gr=1&b=(10791381.798845354,3109633.7229259782;14055506
.524756543,6852939.549588665)&&pn=0&auth=SV3aQL689PMv9fKZaae0D85TyFLzK5xeuxHTxNNBBNVtDpnSCE%40%40B1GgvPUDZYOYIZuVt1cv3
uVtGccZcuVtPWv3Guxtdw8E62qvMuTa4AZzUvhgMZSguxzBEHLNRTVtcEWe1GD8zv7u%40ZPuxtfvAughxehwzJVzPPDD4BvgjLLwWvrZZWuB&device_r
atio=2&tn=B_NORMAL_MAP&nn=0&u_loc=13437544,3654207&ie=utf-8&t=1591773370952'
    return target_url

def startCrawl(target_name):
    target_url = getUrl(target_name)
    return sumData(target_url)

def hotCount(target_name):
    hot_list = []
    target_url = getUrl(target_name)
    data = Crawl(target_url)
    hot_data = data['hot_city']
    for hot_city in hot_data:
        hot_count = int(hot_city[hot_city.find('|')+1: ])
        hot_list.append(hot_count)
    hot_sum = sum(hot_list)
    return hot_sum, hot_data

## change the target_name, returns the stores count
target_name = '优衣库'
try:
    print('Total count: ', startCrawl(target_name))
except:
    print('Hot_city count: ', hotCount(target_name))
```

```
Total count:  1282
```

To provide a convenient view of the brands specially focused on, a scrapper was written to gather the data from JD.com including each brand's sales volume, prices distribution and shop types, with Tableau utilized for better visualization.

In [ ]:
```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from openpyxl import load_workbook
import time
import json
import pandas as pd
import random
import matplotlib.pyplot as plt

options = webdriver.ChromeOptions() # declare a Chrome Driver
options.add_experimental_option('prefs', {'profile.managed_default_content_settings.images': 2}) # set not to display
pictures
browser = webdriver.Chrome(options= options)

url = 'https://www.jd.com/'

def startScraper(keywords):
    data_list = [] # declare a list to store dicts
    browser.get(url) # request url
    browser.find_element_by_id('key').send_keys(keywords) # enter keywords
    browser.find_element_by_id('key').send_keys(Keys.ENTER) # start search
    WebDriverWait(browser, 1000).until(
        EC.presence_of_all_elements_located(
            (By.CLASS_NAME, 'pn-next')
        )
    ) # wait till all the elements all loaded
    all_page = eval(browser.find_element_by_css_selector('span.p-skip em b').text) # get #num of pages
    count = 0 # set a counter
    # loop before the last page
    while True:
        try:
            count += 1
            # wait till all the information loaded
            WebDriverWait(browser, 1000).until(
                EC.presence_of_all_elements_located(
                    (By.CLASS_NAME, 'gl-item')
                )
            )
            browser.execute_script('document.documentElement.scrollTop=10000') # draw the bar to the bottom to load al
l the information
            time.sleep(random.randint(1, 3)) # randomly stop for seconds
            browser.execute_script('document.documentElement.scrollTop=0') # back to the top

            # extract information from label li
            lis = browser.find_elements_by_class_name('gl-item')
            for li in lis:
                # name
                name = li.find_element_by_xpath('.//div[@class="p-name p-name-type-2"]//em').text
                # price
                price = li.find_element_by_xpath('.//div[@class="p-price"]//i').text
                # #num of comments
                comment = li.find_elements_by_xpath('.//div[@class="p-commit"]//a')
                if comment:
                    comment = comment[0].text
                else:
                    comment = None
                # shop name
                shop_name = li.find_elements_by_class_name('J_im_icon')
                if shop_name:
                    shop_name = shop_name[0].text
                else:
                    shop_name = None
                # shop type
                shop_type = li.find_elements_by_class_name('goods-icons')
                if shop_type:
                    shop_type = shop_type[0].text
                else:
                    shop_type = None

                # declare a dict to store data
                data_dict = {}
                data_dict['name'] = name
                data_dict['price'] = price
                data_dict['comment'] = comment
                data_dict['shop_name'] = shop_name
                data_dict['shop_type'] = shop_type

                data_list.append(data_dict)
                #print(data_dict)
        except Exception as e:
            continue
        if count == all_page:
            break
        # find element for next page and click
        fp_next = browser.find_element_by_css_selector('a.fp-next')
        browser.execute_script('document.documentElement.scrollTop = 10000')
        fp_next.click()
    return data_list

# prune the 'comments' to 'sales'
def prune_sales(comments):
    if comments == None:
        return int(0)
    elif comments == '':
        return int(0)
    elif comments[-1] != '+':
        return int(comments)
    elif comments[-2] == '万':
        return int(float(comments[:-2]) * 1e4)
    else:
        return int(comments[:-1])

# save data in excel
def main(keywords):
    result = startScraper(keywords)
    result_df = pd.DataFrame(result)
    result_df['sales'] = result_df['comment'].apply(prune_sales)
    with pd.ExcelWriter('sku_sells.xlsx', engine = 'openpyxl') as writer:
        writer.book = load_workbook('sku_sells.xlsx')
        result_df.to_excel(writer, sheet_name = keywords)

keywords = "雅诗兰黛" # enter keywords
main(keywords)
```