Сбор и разметка данных

Урок 5. Scrapy

Домашнее задание

1. Найдите сайт, содержащий интересующий вас список или каталог. Это может быть список книг, фильмов, спортивных команд или что-то еще, что вас заинтересовало.
2. Создайте новый проект Scrapy и определите нового паука. С помощью атрибута start_urls укажите URL выбранной вами веб-страницы.
3. Определите метод парсинга для извлечения интересующих вас данных. Используйте селекторы XPath или CSS для навигации по HTML и извлечения данных. Возможно, потребуется извлечь данные с нескольких страниц или перейти по ссылкам на другие страницы.
4. Сохраните извлеченные данные в структурированном формате. Вы можете использовать оператор yield для возврата данных из паука, которые Scrapy может записать в файл в выбранном вами формате (например, JSON или CSV).
5. Конечным результатом работы должен быть код Scrapy Spider, а также пример выходных данных. Не забывайте соблюдать правила robots.txt и условия обслуживания веб-сайта, а также ответственно подходите к использованию веб-скрейпинга.

Решение

https://github.com/allseenn/api/tree/main/05.Tasks

1. Выбор сайта

Сайт выбрал, из предложенных и рассмотренных преподавателем на семинаре - hh.ru. И это не простое копирование кода с экрана монитора! Во первых, исходники не выложены к материалам урока, во вторых, мощность и функционал моего решения масштабнее.

Данная работа с hh.ru охватывает:

- вопросы подключения сторонних модулей к фреймворку Scrapy;
- использование сторонних служб в пайплайне с мидлваре;
- расширенный парсинг страниц, включающий полное описание вакансий;
- реализация, практически готового решения, для кадровых агенств, либо для собственных коммерческих и не очень целей.

2. Проект Scrapy

Создание любого проекта Scrapy и его запуск из командной строки не зависит от среды разработки и виртуального окружения. Но, настройка, отладка и запуск в системной (общей) среде пайтон-окружения с Visual Studio Code отличается, от работы в виртуальном окружении PyCharm.

Виртуальное окружение пайтона, отличный инструмент для изоляции приложения от пакетов и зависимостей соседних программ. Но, есть и свои недостатки: размер занимаемого дискового пространства, дополнительно время на установку и повышенный трафик. Виртуальное окружение пайтона, эквивалентно статической компиляции в C/C++, когда все необходимые зависимости и библиотеки компилируются вместе, получается независимый, но очень большой файл (пакет). Но, есть в СИ и динамическая компиляция приложения, позволяющая добиться его наименьшего размера. Динамическая сборка пакетов (*.deb, *.rpm, etc.) используется большинством дистрибутивов Linux.

Одной из проблем, в результате использования общего (системного) окружение пайтона, была фиксация корневой директории scrapy. При запуске из командной строки инструкции типа `scrapy crawl hhru` проблем не возникает, а вот в процессе отладки, пришлось повозиться с решением, а именно в файле runner.py прописать следующий код:

```
import os
os.chdir(os.path.dirname(os.path.abspath(__file__)))
```

В результате, корневой директорией становится, та из которой запущен сам файл, в наше случае runner.py.

3. Парсинг

Модуль hhru.py описывает класс основного паука. Тут, реализовал вывод в консоль запросов пользователю:

- Название искомой вакансии
- Ввод индекса региона для поиска

Запрошенные от пользователя данные используются в параметрах передаваемых в url. В результате чего можно не меняя код приложения осуществлять парсинг по разным профессиям и регионам.

В hhru.py заложил основную логику парсинга, в результате чего собирается по 12 элементов (items):

- id - для исключения дублирования записей в облачной базе Atlas Mondodb, "выдираю" уникальный id из ссылки на вакансию;
- vacancy_title - наименование вакансии;
- vacancy_salary - оклад, парсится в несколько этапов: очистка от "висячих" пробелов, переименование символа ₽, в рубл., очистка от спец символов, с сохранением сум в целочисленном значении. На выходе получается список, который может содержать не только размер оклада, но и условия по его получению. Если никакой информации об окладе нет, с помощью исключений создается пустой список.
- vacancy_experience - получаем список, который содержит помимо количества стажа в годах, еще условия работы (удаленная, полная занятость и т.д.).

- company_name - Название компании, в html коде часто дублируется в одном и том же теге, типа ООО "РогаИКопыта", ООО "РогаИКопыта", более того, название компании может быть указано как в сокращенном, так и в полном наименовании, на русском языке и одновременно на иностранном. Для сокращения информации и упорядочения применяю метод преобразования списка в множество и обратно, в результате остаются только уникальные значения.
- company_link - строка со ссылкой на профиль компании.
- company_rate - вещественное число, с рейтингом компании, на основе отзывов.
- company_address - список адресов, компании. Она может иметь несколько адресов и филиалов, часто происходит дублирование типа: Москва, Москва. Для сокращения объема и достижения уникальности информации, применяю метод со множествами.
- description - самый емкий элемент, в начале хотел объединить все в одну большую строку, но при просмотре конечного файла json, либо при запросе в MongoDB, описание смотрится не читабельно, поэтому решил использовать список строк, что позволило добиться читаемости даже в выводе из базы либо в json-файле.
- vacancy_date - дата парсится, чистится от мусора и преобразуется в формат даты.
- vacancy_link - строка, содержит ссылку в интернете на данную вакансию.
- vacancy_skills - список тэгов с требуемыми навыками.

4. Данные

В процессе парсинга, настроил scrapy для отправки обработанных данных, на облачную службу Atlas MongoDB. Но, для выгрузки нужной коллекции из базы данных в json-файл написал небольшой скрипт.

В результаты скрейпинга, было обработано и сохранено в базу около 800 вакансий, для этого фреймворком было совершено около 1000 запросов к сайту hh.ru.

Как правило, такие крупные проекты испытывают колоссальную нагрузку на свои вычислительные мощности, от большого наплыва посетителей в поисках или размещениях вакансий по все стране.

Не мало хлопот, сайтам наподобие hh.ru доставляют скрейперы. Особенно после семинаров, в которых показано как обрабатывать их сайт.

Естественно, что у hh.ru стоит защита против скрейпинга. После первого и короткого, сканирования, мой ip-адрес был заблокирован.

Для успешной миссии по скрейпингу тысячи вакансий, хорошо подойдут "прокси карусели".

5. Результат

Прокси карусель организована с помощью модуля *scrapy_rotating_proxies*.

Для получения сотни прокси серверов, воспользовался одним из многочисленных сайтов https://proxyscrape.com/, доступ к которому возможен только по ВПН, либо через другой прокси, т.к. для нашей страны есть ограничения. Тем не менее, сайт дает 30-дневный бесплатный доступ к сотням прокси адресов из америки и канады. Не смотря на блокировку страны, доступ к их api свободный.

Существует множество бесплатных сайтов и репозиториев на гитхабе, где выложены в открытый доступ списки прокси-серверов. Но, как показывает практика, такие серверы часто перегружены, либо не работают.

Сайты, с бесплатными триалами, как правило, в рекламных целях дают гарантированно рабочие прокси. Кстати написал скрипт, загружающий список и проверяющий прокси на доступность.

Используя модуль *scrapy_rotating_proxies*, необходимости в самописных скриптах нет. Он может, проверять доступность серверов и пускать трафик через рабочие сервера. Он, самостоятельно отслеживает, не только доступность, но и контролирует карусель и следит за возвращаемым кодом от скрейпируемого сайта, а в случае не положительных кодов в ответе, контролирует скорость вращения на карусели.

Для установки модуля используем команду:

```
pip install scrapy_rotating_proxies
```

Далее, в файле настроек, необходимо добавить следующие строки:

```
DOWNLOADER_MIDDLEWARES = {
   'rotating_proxies.middlewares.RotatingProxyMiddleware': 100,
   'rotating_proxies.middlewares.BanDetectionMiddleware': 100,
}
```

Там же, необходимо прописать путь до файла со списком прокси:

```
ROTATING_PROXY_LIST_PATH = '../../../proxy_list.txt'
```

Использовал всего 100 прокси, можно больше для масштабных проектов. Выставил относительно щадящие настройки:

```
USER_AGENT = ua().random
CONCURRENT_REQUESTS = 32
DOWNLOAD_DELAY = 0.5
```

В результате, через час вращения паука на карусели с прокси, база из 800 вакансий была собрана обработана и залита в облачную базу Atlas MongoDB:

- Фотообразец данных
- JSON-файл вакансий