## Московский авиационный институт (национальный исследовательский университет)

### Институт информационные технологии и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Информационный поиск»

Студент: Е.М. Стифеев Преподаватель: А.А. Кухтичев Группа: M8O-109M-21

Дата: 11.10.21

Оценка: Подпись:

#### Лабораторная работа №1 «Добыча корпуса документов»

Необходимо подготовить корпус документов, который будет использован при выполнении остальных лабораторных работ:

- Скачать его к себе на компьютер. В отчёте нужно указать источник данных.
- Ознакомиться с ним, изучить его характеристики. Из чего состоит текст? Есть ли дополнительная мета-информация? Если разметка текста, какая она?
- Разбить на документы.
- Выделить текст.
- Найти существующие поисковики, которые уже можно использовать для поиска по
- выбранному набору документов (встроенный поиск Википедии, поиск *Google* с использованием ограничений на *URL* или на сайт). Если такого поиска найти невозможно, то использовать корпус для выполнения лабораторных работ нельзя!
- Привести несколько примеров запросов к существующим поисковикам, указать недостатки в полученной поисковой выдаче.

В результатах работы должна быть указаны статистическая информация о корпусе:

- Размер «сырых» данных.
- Количество документов.
- Размер текста, выделенного из «сырых» данных.
- Средний размер документа, средний объём текста в документе.

#### 1. Описание

Для выполнения поставленной задачи требуется написать веб-скрапер. После длительных размышлений над тематикой документов было принято решение «обкачать» сайты с кинотематикой. И, конечно же, в качестве первого кандидата был выбран крупнейший сборник подобных документов ру-сегмента интернета *kinopoisk* (https://www.kinopoisk.ru/).

Рассмотрим структуру страницы о фильме на примере <a href="https://www.kinopoisk.ru/film/64021/">https://www.kinopoisk.ru/film/64021/</a> (х/ф «Зомби по имени Шон»). Открыв её в браузере, наблюдаем следующую интересующую нас информацию для скрапинга:

- Название фильма («Зомби по имени Шон»);
- Альтернативное название («Shaun of the Dead»);
- Год производства (2004);
- Средняя оценка пользователей (7.4);
- Страна/страны производства фильма (Великобритания, Франция);
- Режиссёр/режиссёры (Эдгар Райт);
- Сценарист/сценаристы (Саймон Пегг, Эдгар Райт);
- Актёры (Саймон Пегг, Ник Фрост, ...);
- Жанры (ужасы, комедия);
- Описание («В жизни Шона все идет наперекосяк ...»);
- Рецензии («Пересматриваю эту пародию ритуально ...»).

Теперь нужно каким-то образом найти ссылки на все фильмы из базы. Для этого была найдена карта сайта <a href="https://www.kinopoisk.ru/sitemaps/sitemap.xml">https://www.kinopoisk.ru/sitemaps/sitemap.xml</a> (ссылка на неё указана в <a href="https://www.kinopoisk.ru/robots.txt">https://www.kinopoisk.ru/robots.txt</a>), откуда был извлечён список ссылок на все фильмы, хранящиеся на сайте. Получив такой список, можно «распарсить» каждую ссылку, выделив нужные сведения из html-кода, совмещая такие техники как:

- Получение кода html-кода страницы (Scrapy, Selenium)
- Исследование кода страницы через встроенный в браузер (в моём случае *Yandex Browser*) инспектор сайта, который помогает понять, какой блок *html*-кода, за какой элемент страницы отвечает.
- Средства анализа древовидной структуры страницы (*BeautifulSoup*) с целью извлечения нужных тэгов.
- Сохранение информации в локальный файл (в моём случае *jsonlines*).

В процессе выполнения работы автор столкнулся с такими проблемами, как:

#### • Динамически подгружающая секциями с отзывами.

Сначала был написан скрипт, использующий фреймворк *Selenium* для выполнения *java script* а на странице, работающий по принципу: загрузить страницу, проскроллить в её конец, подождать пока на странице на появится нужный блок с отзывами (по появлению определённого тэга) и тогда извлечь отзывы из блока, либо пока не пройдёт, например, 5 секунд, и тогда завершить обработку страницы без отзывов.

Затем автор выяснил по карте сайта, что есть ссылки на страницы, на которых находятся только отзывы <a href="https://www.kinopoisk.ru/film/64021/reviews/">https://www.kinopoisk.ru/film/64021/reviews/</a> (64021 — уникальный номер фильма). В связи с чем было принять решение обрабатывать по две страницы на фильм (фильм + отзывы) с секундной задержкой.

#### Капча.

При её появлении было принято решение пропускать обработку таких страниц, занося ссылку в отдельный файл с целью ручного распознавания в будущем (таких страниц было 1% от всех). Также можно воспользоваться таким инструментом как *tesseract* [1] для автоматической обработки (это также требует ручной обработки определённого количества изображений).

Дополнительно была обкачана новостная лента <a href="https://www.kinopoisk.ru/media/">https://www.kinopoisk.ru/media/</a>, торрент-трекер <a href="http://filmplace.ru/">http://filmplace.ru/</a> и портал по японской мультипликации <a href="https://shikimori.one/">https://shikimori.one/</a>.

#### 2. Исходный код

#### Структура проекта

- film\_scraper
  - o film\_scraper
    - сартсћа (файлы со ссылками на страницы с капчей)
      - sitemap\_captcha\_links1.txt
      - sitemap\_captcha\_links2.txt
    - jsons (выходные файлы)
      - films1.jsonlines
      - films2.jsonlines
    - sitemaps (файлы со ссылками для обработки пауком)
      - sitemap\_film\_review1.txt
      - sitemap\_film\_review2.txt
      - sitemap\_film1.txt
      - ...
      - sitemap\_film18.txt
    - spiders (пауки)
      - \_\_init\_\_.py
      - FSpyder.py
    - \_\_init\_\_.py
    - items.py
    - middlewares.py
    - pipelines.py
    - setting.py (глобальные настройки)
  - o scrapy.cfg
  - o chromedriver.exe

#### Запуск

Итоговый «паук», скачанный с <a href="https://github.com/Stifeev/Information-retrieval/tree/main/JIP1">https://github.com/Stifeev/Information-retrieval/tree/main/JIP1</a>, требует настройки абсолютного пути до рабочей папки в файле ./film\_scraper/film\_scraper/settings.py:

Запуск осуществляется командой ниже из директории ./film\_scraper/spiders:

## \$ scrapy runspider FSpider.py

Также требуются предустановленные *Python*-библиотеки (*Python* 3-й версии):

- scrapy;
- selenium;
- re;
- *bs4*;
- *jsonlines*;
- urllib.

# Структура FSpyder.py

| Сигнатура                                       | Описание                                   |
|---|--|
| <pre>def extract_key(url)</pre>                 | Извлечь уникальный номер фильма            |
|   | из ссылки на него                          |
| <pre>def find_wrap(search)</pre>                | Безопасный механизм обработки              |
|   | поиска в <i>html</i> -коде (возврат пустой |
|   | строки, если поиск неудачный)              |
| <pre>def fetch_headers(page, item)</pre>        | Извлечь основную информацию из             |
|   | страницы (всё, кроме отзывов) в            |
|   | объект item                                |
| <pre>def fetch_reviews(spider, item, url,</pre> | Извлечь отзывы в объект <i>item</i> .      |
| review_count)                                   | Функция умеет обрабатывать случаи,         |
|   | когда фильм имеет больше отзывов,          |
|   | чем REVIEWS_OFFSET (200 по                 |
|   | умолчанию) через обработку                 |
|   | нескольких страниц                         |
| <pre>class FSpider(Spider)</pre>                | Класс паука                                |
| <pre>definit(self, *args, **kwargs)</pre>       | -Извлечение ключей всех записей из         |
|   | текущего <i>json</i> -файла в множество    |
|   | (set) film_ids для предотвращения          |
|   | добавление неуникальных записей;           |
|   | -Извлечение списка ссылок на               |
|   | фильмы для обработки из файла и            |
|   | сравнение их с ключами;                    |
|   | -Очистка файла с ссылками на               |
|   | страницы с капчей;                         |
|   | -Инициализация веб-драйвера                |
| <pre>def parse(self, response)</pre>            | Основная функция обработки                 |
|   | страницы. Извлечение <i>item</i> 'a.       |
| <pre>def closed(self, reason)</pre>             | Обработка остановки паука                  |

#### 3. Выводы

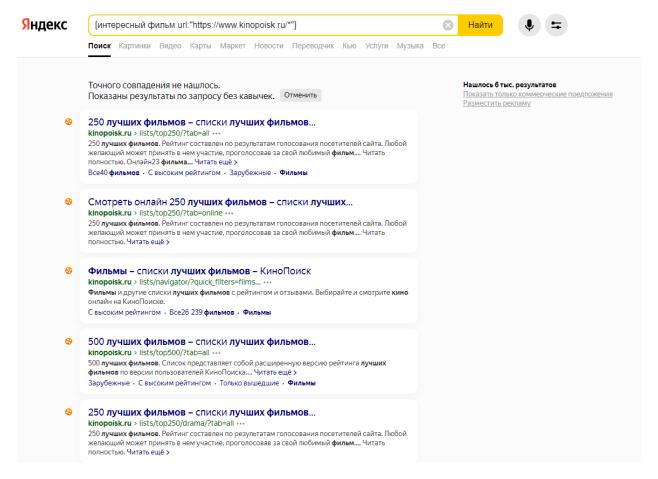
На данный момент удалось обработать несколько сайтов: www.kinopoisk.ru, filmplace.ru и shikimori.one.

| Сырые данные                        |               |
|-------------------------------------|---------------|
| Общее число обработанных страниц:   | ≈700'000      |
| Общий объём обработанных страниц:   | ≈120 Gb       |
| Отфильтрованные данные              |               |
| Общий объём файлов:                 | 3.15 Gb       |
| Общее число символов:               | 1'558'003'484 |
| Общее число документов:             | 170477        |
| Среднее число символов в документе: | 9139          |
| Средний объём документа:            | 19.5 Kb       |

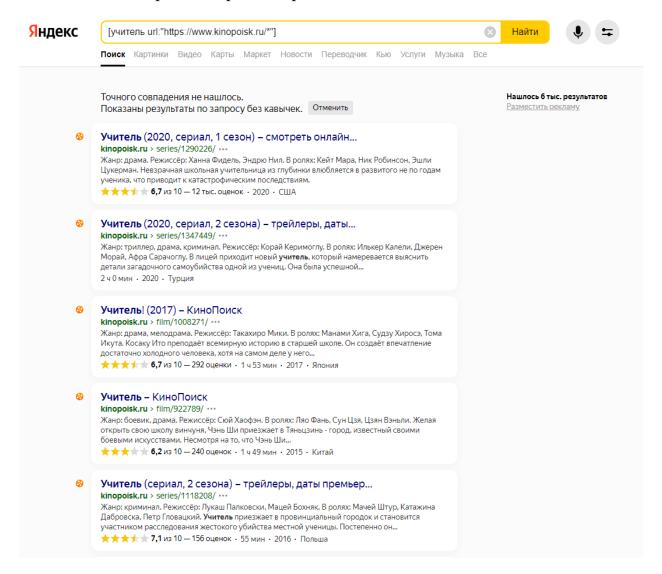
Ссылка на корпус документов: <a href="https://cloud.mail.ru/public/ELAx/f9NqPZ12r">https://cloud.mail.ru/public/ELAx/f9NqPZ12r</a>.

#### Примеры запросов:

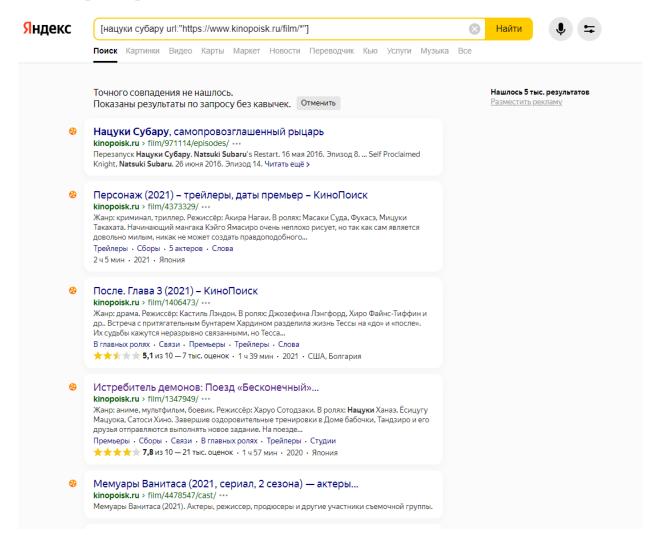
• Интересный = лучший



#### • А ещё это фамилия режиссёра



#### • Герой сериала



В ходе выполнения лабораторной работы я научился анализировать и скачивать *html*-страницы в автоматическом режиме, с целью извлечения нужных сведений. Это может пригодится для сбора статистики по сайту, создание агрегаторов, таких как, например, агрегатора цен на авиабилеты, поиска уязвимостей, получение корпуса документов для машинного обучения и прочее.

## Список литературы

[1] Райан Митчелл *Современный скрапинг веб-сайтов с помощью Python*. 2-е межд. издание. — СПб.: Питер, 2021 — 528 с. (ISBN 978-5-4461-5)