**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: | М8О-109М-21 |
| Дата: | 11.10.21 |
| Оценка: |  |
| Подпись: |  |

**Москва, 2021**

# 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.

# 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 <https://www.kinopoisk.ru/film/64021/> (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 <https://www.kinopoisk.ru/sitemaps/sitemap.xml> (ссылка на неё указана в <https://www.kinopoisk.ru/robots.txt>), 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 json).

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 <https://www.kinopoisk.ru/film/64021/reviews/> (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* для автоматической обработки (это также требует ручной обработки определённого количества изображений).

В дальнейшем планируется пополнить список подобных сайтов и провести с ними аналогичную работу по извлечению нужных документов. Для текущего сайта можно ещё обработать новостную ленту.

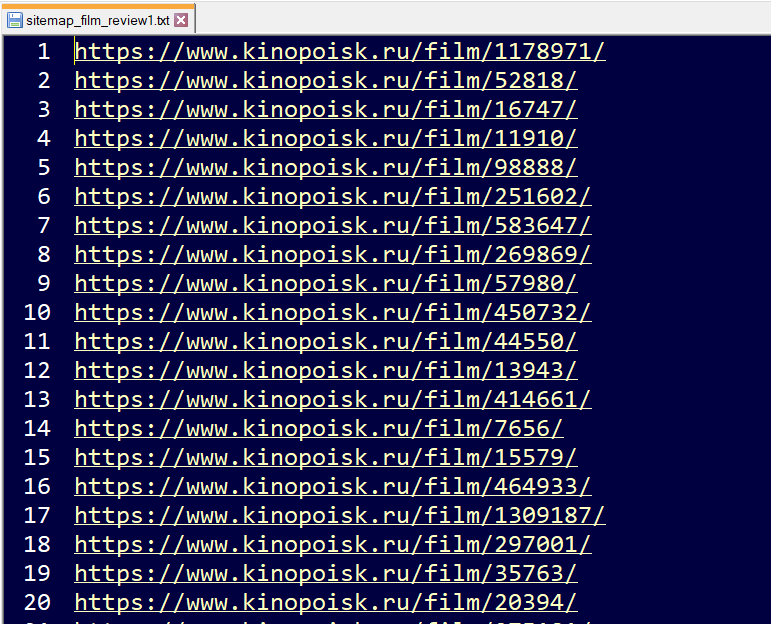
# Исходный код

Структура проекта:

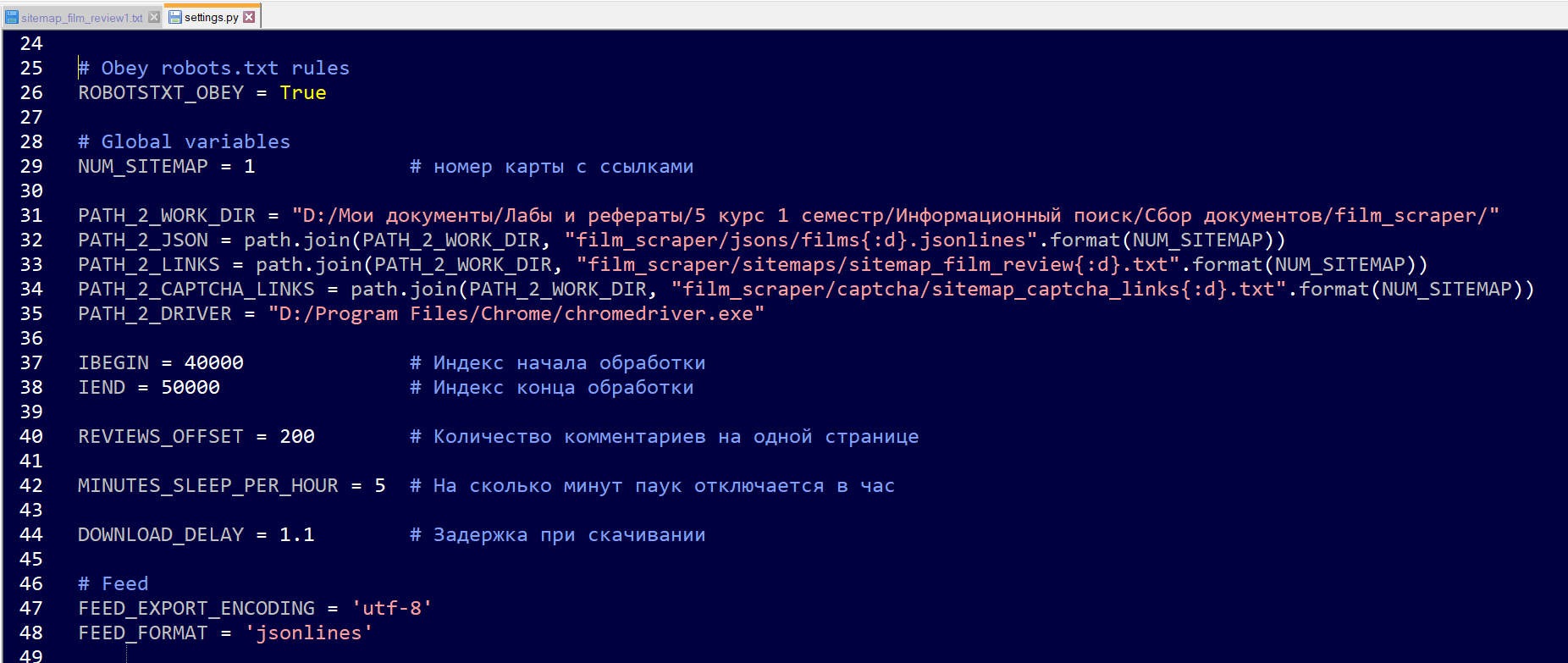
* film\_scraper
  + film\_scraper
    - captcha (файлы со ссылками на страницы с капчей)
      * sitemap\_captcha\_links1.txt
    - jsons (выходные файлы)
      * films1.jsonlines
    - sitemaps (файлы со ссылками для обработки пауком)
      * sitemap\_film\_review1.txt
      * sitemap\_film\_review2.txt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

Итоговый «паук» требует следующие этапы настройки:

* Нужно создать txt-файл со списком ссылок для обработки (\film\_scraper\film\_scraper\sitemaps\sitemap\_film\_review1.txt)



* Далее, нужно настроить параметры в файле \film\_scraper\film\_scraper\settings.py:



* Запустить «паука» командой с указанием пути выходного файла:

$ scrapy runspider spiders/FSpider.py -o jsons/films1.jsonlines

Структура FSpyder.py:

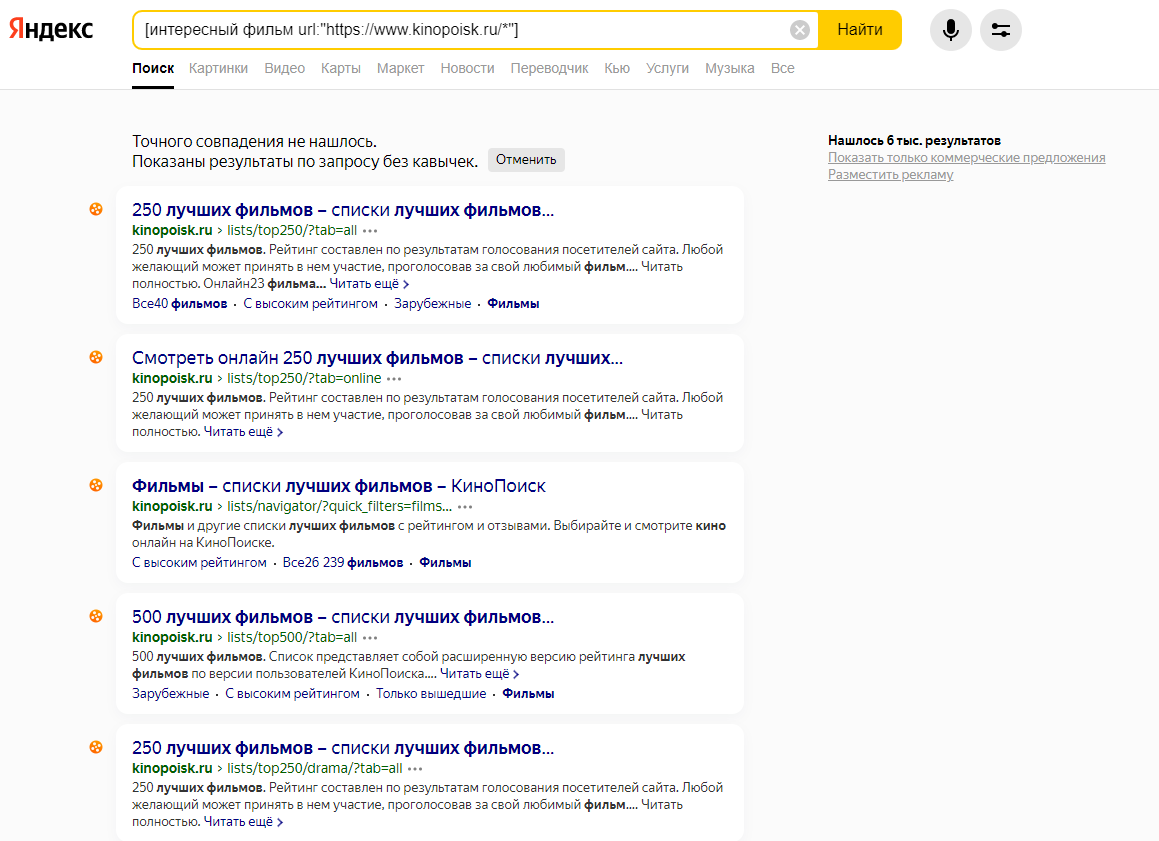
|  |  |
| --- | --- |
| Сигнатура | Описание |
| **def** extract\_key(url) | Извлечь уникальный номер фильма из ссылки на него |
| **def** find\_wrap(search) | Безопасный механизм обработки поиска в *html*-коде (возврат пустой строки, если поиск неудачный) |
| **def** fetch\_headers(page, item) | Извлечь основную информацию из страницы (всё, кроме отзывов) в объект item |
| **def** fetch\_reviews(item, driver, url,  review\_count, logger,  captcha\_fp) | Извлечь отзывы в объект item. Функция умеет обрабатывать случаи,  когда фильм имеет больше отзывов,  чем REVIEWS\_OFFSET (200 по умолчанию) через обработку нескольких страниц |
| **class** FSpider(Spider) | Класс паука |
| **def** \_\_init\_\_(self, \*args, \*\*kwargs) | -Извлечение ключей всех записей из  текущего json-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 |
| **def** parse(self, response) | Основная функция обработки страницы. Извлечение item’а. |
| **def** closed(self, reason) | Обработка остановки паука |

# Выводы

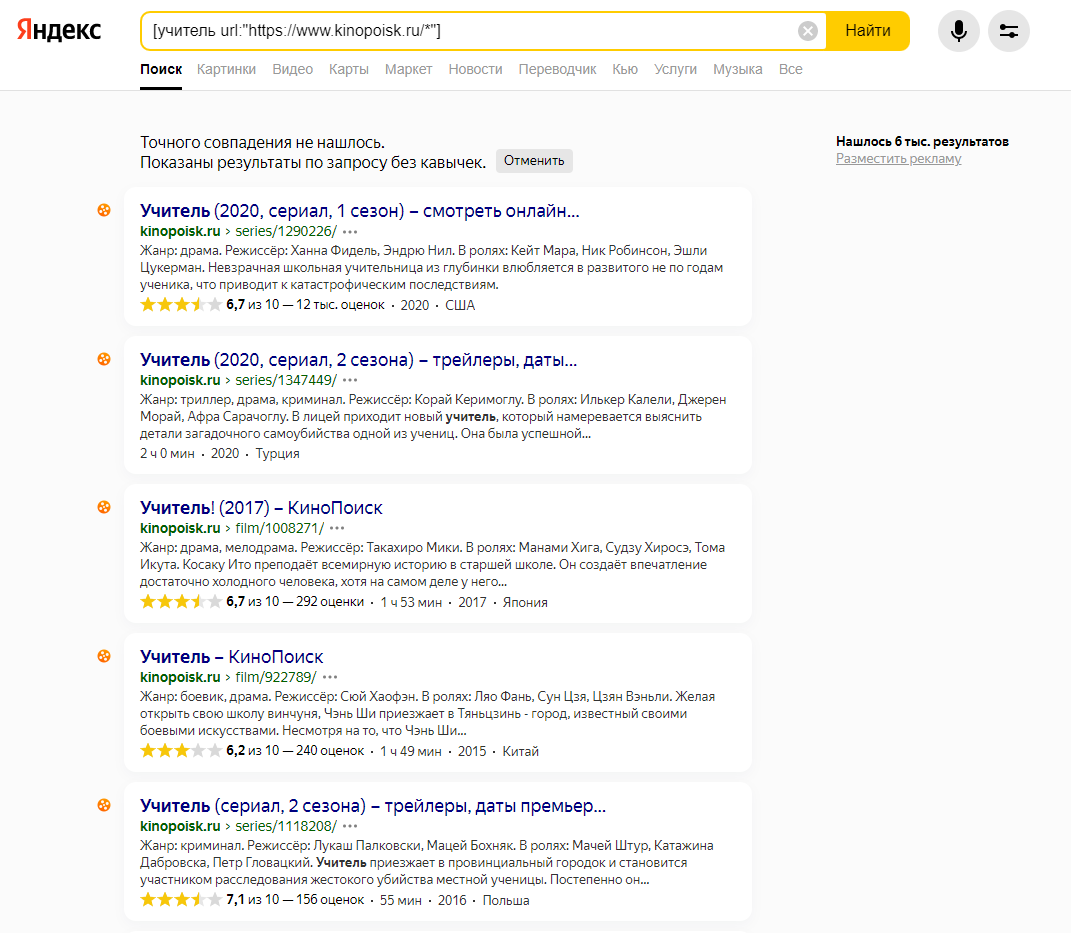
На данный момент удалось обработать 60492(фильмы)+60492 (рецензии)=120984 страниц с фильмами и рецензиями, суммарным объёмом > 31 ГБ. Количество полученных документов соответственно составляет 60492, объёмом 1.5 ГБ, суммарной длиной в символах 807’172’519. Средний размер документа в символах: 13343. Средний размер документа: 26 Кбайт.

Примеры запросов:

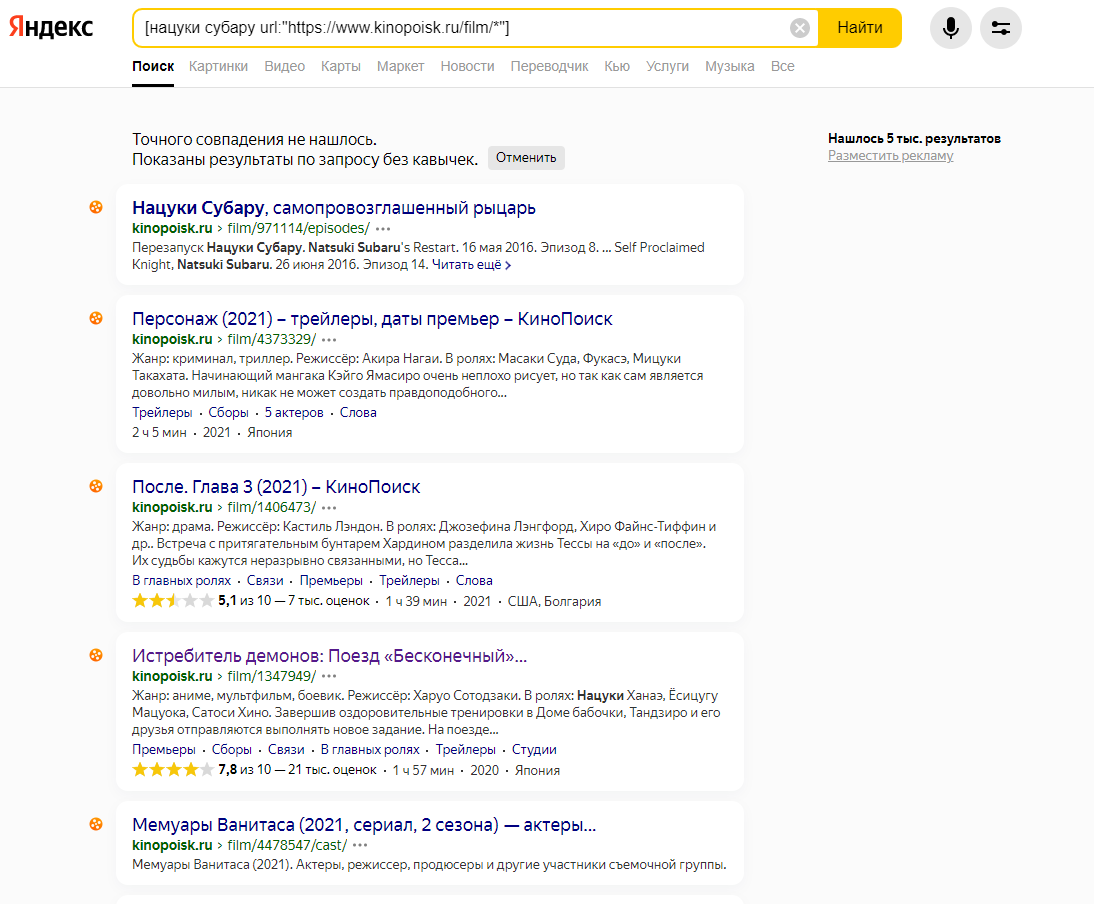
* Интересный = лучший



* А ещё эта 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)