УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Зорана Гајић

САВРЕМЕНИ АЛАТИ ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА

мастер рад

Београд, 2023.

Ментор:

др Милена Вујошевић Јаничић, ванредни професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Ана Аничић, ванредни професор
University of Disneyland, Недођија

др Лаза Лазић, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 15. јануар 2016.

Породици

Наслов мастер рада: Савремени алати за прикупљање података са веб-страница

Резиме:

Кључне речи: прикупљање података са веб-страница

Садржај

Глава 1

Увод

TODO:

Глава 2

Прикупљање података са веб-страница

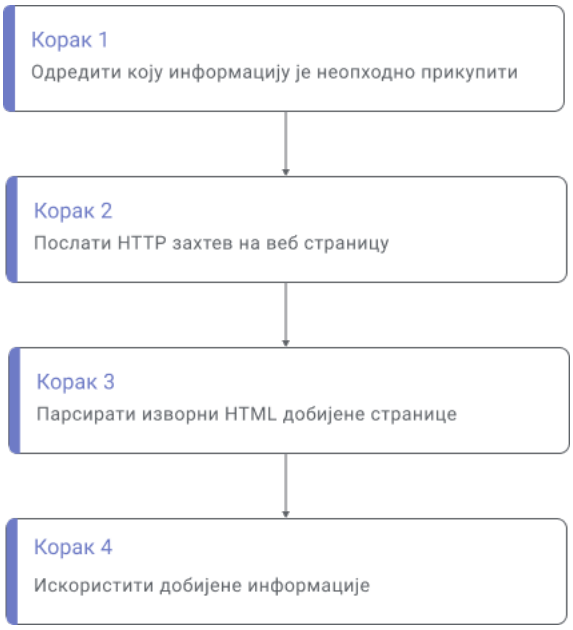
Веб (енг. WWW, World Wide Web [1]) је тренутно највећи извор података у историји човечанства и састоји се углавном од неструктурираних података, које су можда компликоване за прикупљање [4]. Већина веб-сајтова не дозвољавају копирање или преузимање података, а ручно копирање података може потрајати данима и недељама.

Веб скрејпинг представља аутоматизовани процес у којем се подаци издвајају са различитих веб-страница и чувају се локално у структурираном формату за тренутну употребу или анализу која ће се касније извршити. Процес прикупљања података може бити написан на различитим програмским језицима, од којих су најпопуларнији: Пајтон, Јава и Руби.

Како се врши прикупљање података? На приказаној схеми 2.1 се може пронаћи одговор на постављено питање. Дакле, најпре је неопходно пронаћи веб-страницу погодну за прикупљање података (више о томе у секцији 2.1) и одредити које тачно информације треба прикупити. Затим је неопходно послати *HTTP* (енг. HTTP, Hypertext Transfer Protocol [2]) захтев на жељену веб-страницу и преузети изворни HTML странице. Пре самог парсирања HTML-а је неопходно пронаћи најбољи начин за индексирање жељених елемената, затим парсирати HTML и извршити неопходну акцију са добијеним информацијама, поштујући смернице које ће бити дате у секцији 2.1.

[1] Светска мрежа, познатија као Веб, систем је међусобно повезаних, хипертекстуалних докумената који се налазе на интернету.

[2] HTTP је мрежни протокол који припада слоју апликације ОСИ референтног модела, представља главни и најчешћи метод преноса информација на Вебу.



Слика 2.1: Веб скрејпинг кораци

2.1 Изазови

Веб скрејпинг се сматра одличним алатом за стицање увида у податке, али такође треба обратити пажњу на правне аспекте, како не би дошло до легалних проблема. Оно што са сигурношћу можемо да тврдимо јесте да ниједан веб-сајт не жели да допусти крађу података. Можда неки од њих немају имплементиране механизме одбране, али они који ипак користе анти-скрејпинг мере могу да доведу до блокирања прикупљања података, јер не верују у отворени приступ подацима.

Да би прикупљање података било успешно, морамо да будемо сигурни да су добијени подаци квалитетни. Самим тим је неопходно избегавати веб-сајтове који могу да угрозе квалитет података. То су углавном веб-сајтови који имају превише неисправних линкова, из разлога што управо преко тих линкова прелазимо цео веб-сајт и прикупљамо податке. Такође би требало избећи веб-сајтове са веома динамичним кодирањем, из разлога што се изворни HTML стално мења и онда код за парсирање постаје ирелевантан.

Када су у питању већи пројекти, са великим базама података, чест је изазов складиштења података, који може да се реши коришћењем на пример AWS, Amazon Web Services [3]. Проблем динамичности веб-сајта можемо

[3] `https://aws.amazon.com/?nc1=h_ls`

да заобиђемо приказивањем веб-странице у претраживачу без заглавља (енг. Headless Chromium [6]), који у суштини омогућава покретање претраживача у окружењу сервера. Један од већих изазова је пријављивање корисника на веб-страници, али је то решиво уз помоћ библиотека као што су Selenium и Scrapy што ће бити приказано касније у раду.

У наставку су описане најчешће праксе за прикупљање података, које ћемо применити у оквиру имплементације у секцији 3.2:

1. Поштовати robots.txt фајл.

   Овај фајл је од изузетне важности јер представља политику веб-сајта. Најчешће се проналази на нивоу основног директоријума, као на пример `http://example.com/robots.txt`. Уколико фајл садржи линије попут ових приказаних у наставку, то значи да веб-сајт не жели да се прикупљају подаци са њега

   ```
   User-agent: *
   Disallow:/
   ```

2. Мењање корисничког агента.

   Сваки HTTP захтев у заглављу шаље корисничког агента (енг. User agent). Коришћењем овог подешавања веб-сајт идентификује претраживач који му приступа: његову верзију и платформу. Уколико користимо истог корисничког агента у сваком захтеву, веб-сајт може лако да открије да је у питању аутоматизовани приступ страници.

3. Одговорно коришћење прикупљених података.

   Мора да се сноси одговорност за прикупљене податке и водити рачуна како и у које се сврхе користе, а да се не крши закон о ауторским правима.

4. Учесталост прикупљања података свести на минимум.

   Веб-страница за анти-скрејпинг меру може имати одређен интервал фреквенције за ботове. Да би се ово избегло углавном је довољно да се између два HTTP позива допусти кашњење од 10 секунди.

5. Не пратити увек исти образац прикупљања података, већ додати радње које ће оставити утисак понашања човека као што су: покрети миша, клик на насумични линк, итд.

6. Прикупљати податке када најмањи број посетилаца користи Веб-сајт. Ови сати се могу идентификовати геолокацијом одакле потиче саобраћај на сајту. Такође у ове сате можемо да очекујемо добијање резултата доста брже него у сате када је највећа посећеност.

7. Не прикупљати већ прикупљене податке.

8. Мењање ИП адресе и прокси услуга (енг. Proxy Services)

   Користићемо ротирајући прокси, који представља прокси сервер који свакој конекцији додељује нову ИП адресу из скупа проксија.

2.2 Примењене технологије

TODO: (HTML, Regex, XPath, Тагови, CSS Селектори)

Hyper Text Markup Language

TODO:

Regex

TODO:

XPath

XPath (енг. XML Path Language) [1] представља флексибилан начин адресирања на различите делове XML (XML, Extensible Markup Language [4]) документа и због тога може да се користи као начин навигације кроз DOM (DOM, Document Object Model [5]) било ког документа језика сличног XML користећи *XPath израз* (енг. XPathExpression [6]). Садржи преко 200 уграђених функција

[4]XML представља проширив (мета) језик за означавање (енгл. markup)

[5]DOM или објектни модел документа представља хијерархијски приказ структуре Веб-сајта.

[6]XPath израз дефинише образац за одабир скупа чворова.

и дефинисан је од стране WWW конзорцијума. У овом раду ће се XPath користити као начин за одабир елемената са HTML страница.

XPath синтакса

XPath користи изразе путање за избор чворова у XML документу. Чвор се бира праћењем путање или корака. Корисни примери израза путања су наведени у наставку:

- `//h2`

  прикупља све *h2* елементе

- `//div//p`

  прикупља све *параграф* (енг. p) елементе унутар блока *div*

- `//ul/li/a`

  прикупља све линкове унутар неуређених листи

- `//ol/li[2]`

  проналази други елемент уређене листе

- `//div/*`

  прикупља све унутар неуређених блока *div*

- `//*[@id="id"]`

  проналази елемент са задатим идентификатором

- `//*[@class="class"]`

  проналази све елементе са задатом класом

- `//a[@name or @href]`

  проналази све линкове који имају *name* или *href* атрибут

- `//button[contains(text(),"Send")]`

  проналази сву дугмад која садрже текст *Send*

- `//a[last()]`

  проналази последњи линк

- `//table[count(tr)=1]`

  проналази број редова у табели

- `string()`
  `number()`
  `boolean()`

  Типови конверзија

- `contains()`
  `starts-with()`
  `ends-with()`

  функције са нискама

- `//*`

  цви елементи

- `//section[h1[@id='section-name']]`

  унутар *section* елемента пронаћи све *h1* елементе са жељеним идентификатором

- `//a/text()`

  проналази текст линка

- `./a`

  тачка прикупља тренутни чвор

Тагови

TODO:

CSS Селектори

TODO:

Глава 3

Преглед алата за прикупљање података са веб-страница

TODO: Кратак увод шта ће се користити у поглављу, који програмски језик, које библиотеке, опис сајта, жељени циљеви, препреке са којима ћу се сусрести у раду

3.1 Библиотека BeautifulSoap

BeautifulSoap библиотека [9] представља једну од најједноставнијих за коришћење Пајтон библиотека за имплементацију прикупљања података из HTML и XML текстуалних докумената. Више информација на `https://www.w3.org/TR/xml/`.) датотека преласком кроз DOM. Искључиво ради само статичко прикупљање, које игнорише Јаваскрипт (енг. JavaScript), тј преузима се веб-страница са сервера без помоћи претраживача. За разлику од осталих библиотека које ће се касније у раду разматрати, ова библиотека не може сама да приступи веб-страници, већ јој је неопходна помоћна библиотека. Што значи да јој је главна функција парсирање HTML и XML датотека. Честа је пракса да се динамичан део веб-странице дохвати уз помоћ Selenium пакета [5], а да се парсирање података врши уз помоћ BeautifulSoap пакета.

Инсталација

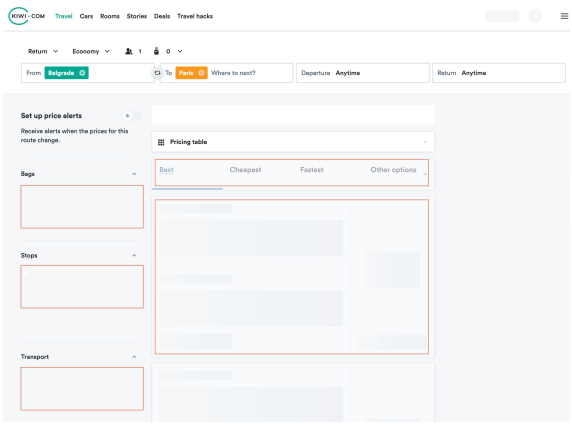
Неопходно је инсталирати *pip3* [1] пакет менаџер и редом *BeautifulSoap*, *requests* [8] и парсер библиотеку *lxml* [3] уз помоћ наведених испод наредби.

```
pip3 install bs4
pip3 install requests
pip3 install lxml
```

Детаљно упутство за инсталацију се може пронаћи у званичној документацији BeautifulSoap библиотеке [9].

Провера динамичности веб-странице

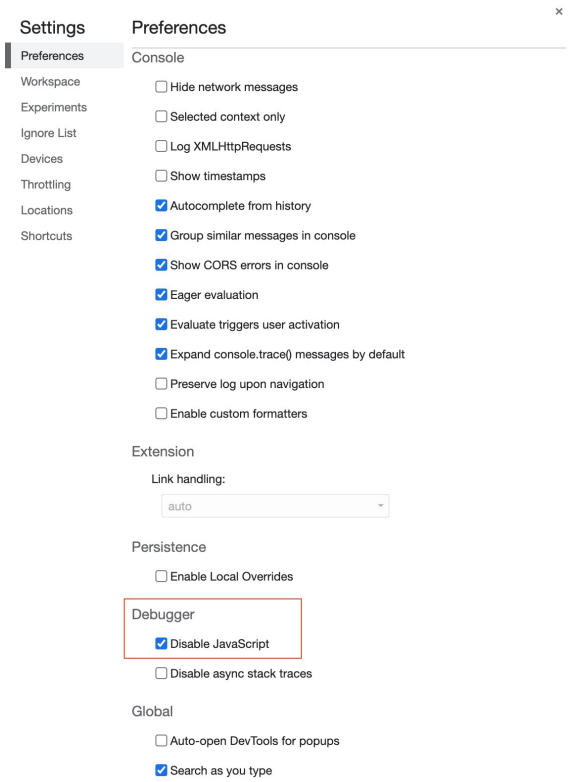
Прва препрека коришћења BeautifulSoap пакета на примеру Киви веб-странице јесте та, да `https://www.kiwi.com/` представља динамичну страницу и да се без Јаваскрипта не може учитати. Самим тим не постоји могућност добијања података конкретно о летовима коришћењем искључиво BeautifulSoap пакета. Изглед веб-странице са угашеним Јаваскриптом је приказан на слици 3.1.



Слика 3.1: Киви веб-сајт са угашеним Јаваскриптом

Да би се видело како конкретна веб-страница зависи од Јаваскрипта, у подешавању коришћеног претраживача је неопходно да се угаси опција коришћења Јаваскрипта и како то урадити на примеру *Хром* (енг. Google Chrome) претраживача је приказано на слици 3.2.

[1] `https://pip.pypa.io/en/stable/`



Слика 3.2: Подешавање претраживача

Прикупљање и парсирање HTML

BeautifulSoap није библиотека за скрејповање података са Веба сама по себи. То је библиотека која омогућава да се ефикасно и лако извуче информација из HTML.

Дакле, за почетак, потребан је HTML документ и у ту сврху се може искористити Пајтонов пакет *requests* који омогућава слање HTTP захтева. Постоји неколико метода од значаја у овом пакету [7]:

- `get(url, params, args)`

  Шаље HTTP GET захтев на наведену веб-адресу (енг. URL, Uniform Resource Locator))

- `post(url, data, json, args)`

  Шаље HTTP POST захтев на наведену веб-адресу

- `put(url, data, args)`

Шаље HTTP PUT захтев на наведену веб-адресу

У наставку је приказан код који преузима веб-страницу. Треба имати у виду да при преузимању странице са Веба, може да се догоди да страница не буде пронађена на серверу (или је дошло до грешке у њеном преузимању) или да сервер није пронађен.

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 website = 'https://www.kiwi.com/en/search/results/belgrade-serbia/
      paris-france'
5 try:
6     response = requests.get(website)
7 except requests.exceptions.RequestException as err:
8     print("Error fetching page")
9     exit()
10
11 content = response.text
```

Даље, неопходно је испарсирати добијену HTML страницу тако што се креира *BeautifulSoap* конструктор који за параметре прихвата HTML страницу и парсер.

Пајтон нуди разне библиотеке за парсирање HTML, од којих су две најзаступљеније: *lxml* [3] и *html.parser* [2]. Сваки пројекат има различите захтеве на основу којих се може одабрати најпогоднији парсер, па је за потребе овог рада одабран lxml парсер из једног простог разлога - брзина. lxml је најбржи парсер HTML страница према званичној документацији BeautifulSoap библиотеке [9]. Једна од мана овог парсера јесте екстерна зависност, али у овом случају то не представља проблем.

Наведени код ниже креира *BeautifulSoap* објекат, који омогућава лак приступ многим корисним информацијама, као што је наслов странице, сви линкови на страници, текст са странице...

```
1 soup = BeautifulSoup(content, 'lxml')
2
3 print(soup.title)
4 # <title>Belgrade-Paris trips</title>
5
6 print(soup.find_all('a'))
```

```
# [<a class="Logo__Link-sc-1uwlkrd-6 kcwlYu" data-test="Logo" href="/
    en/?destination=paris-france&amp;origin=belgrade-serbia">...]

print(soup.get_text())
# Belgrade Paris trips
# TravelCarsRoomsStoriesDealsTravel hacksloadingManage your trips, set
    up price alerts, use...
```

Циљање DOM елемената

У оквиру пакета BeautifulSoap постоје два метода која омогућавају једноставно и брзо филтрирање HTML странице за проналазак жељених информација [7]:

- `find(tag, attributes, recursive, text, keywords)`

  проналази и враћа жељени елемент, где су аргументи редом: назив тага или листа тагова, Пајтон речник атрибута и одговарајућих тагова које садрже било који од тих атрибута, булова вредност која представља да ли се претрага врши по глобалним таговима или и по унутрашњим таговима, подударност на основу текстуалног садржаја ознака, атрибут на основу којих се бира таг

- `findAll(tag, attributes, recursive, text, limit, keywords)`

  проналази и враћа листу жељених елемената, где су аргументи слични наведеним изнад са разликом што је додат аргумент лимит, који представља максималан број подударних елемената

Да би приказане информације биле занимљивије, HTML страница ће бити преузета коришћењем Selenium пакета из разлога што Selenium подржава Јаваскрипт. Детаљније о самом пакету Selenium може да се пронађе у наставку рада 3.2.

Одабрана је страница са летовима на релацији Београд-Малага `https://www.kiwi.com/en/search/results/belgrade-serbia/malaga-spain` и коришћењем Selenium пакета је направљен Пајтон скрипт који ће у одвојени фајл експортовати изворни HTML дате странице. У наставку је приказан код који најпре затвара прозор за *колачиће* (енг. cookies [2]) који се отвара кад

[2]Колачић представља текстуалну датотеку која се чува у веб прегледачу док корисник прегледа неку веб-страницу.

се приступи веб-страници, затим чека 10 секунди да би се страница учитала и цео изворни HTML експортује у одвојени фајл. И овим је решен проблем нединамичности пакета BeautifulSoap.

```
1  import time
2
3  from selenium import webdriver
4  from selenium.common import exceptions
5  from selenium.webdriver.chrome.service import Service
6  from selenium.webdriver.common.by import By
7
8  website = 'https://www.kiwi.com/en/search/results/belgrade-serbia/
       malaga-spain'
9  path = '/usr/local/bin/chromedriver_mac64/chromedriver'
10 service = Service(executable_path=path)
11 driver = webdriver.Chrome(service=service)
12 driver.get(website)
13 driver.maximize_window()
14
15 try:
16     # Close cookies dialog
17     cookie_dialog_close_btn = driver.find_element(By.XPATH, value=".//
       section[@id='cookie_consent']//button[@aria-label='Close']")
18     cookie_dialog_close_btn.click()
19
20     time.sleep(10)
21
22     with open('flights-belgrade-malaga-page-source.txt', 'w') as file:
23         file.write(driver.page_source)
24
25     driver.quit()
26
27 except exceptions.StaleElementReferenceException as e:
28     print(e)
29     pass
```

У наставку ће бити приказан комплетан код који извлачи информације из необрађеног HTML који је претходно био експортован у фајл: датум поласка, информације о одласку, датум повратка, информације о повратку, број ноћи, цена лета.

Најпре је неопходно установити како извући целе блокове са информацијама о лету. Када се детаљно прегледа структура HTML, може се приметити

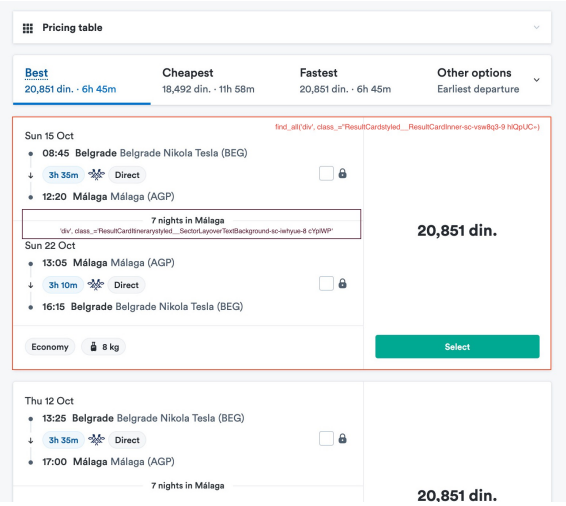
да је сваки лет издвојен у *div* [3] са дугачким именом класе што представља једино по чему се тај блок може идентификовати. Пошто на страници има више летова, најпре се користи функција која проналази све инстанце по потпуној подударности имена класе. На исти начин се извршио проналазак броја ноћи касније. 3.3.

```
all_flights = soup.find_all('div', class_="
    ResultCardstyled__ResultCardInner-sc-vsw8q3-9 hlQpUC")
```

Осим потпуне подударности по имену класе, елемент се може пронаћи по свом јединственом *id* [4]:

```
soup.find(id="specific_id")
```

Сада када су пронађени сви блокови са летовима, могу да се извуку информације о конкретно приказаном лету проласком кроз *for* петљу кроз дате инстанце.



Слика 3.3: Комплетно поклапање класа

```
import re

import pandas as pd
from bs4 import BeautifulSoup

with open('flights-belgrade-malaga-page-source.txt', 'r') as file:
    soup = BeautifulSoup(file, 'lxml')
```

[3] Ознака дефинише поделу или одељак у HTML документу.
[4] id атрибут се користи за одређивање јединственог идентификатора за HTML елемент.

```
8
9       all_flights = soup.find_all('div', class_="
        ResultCardstyled__ResultCardInner-sc-vsw8q3-9 hlQpUC")
10
11      departure_date_regex = re.compile('.*DepartureDate.*')
12      departure_info_regex = re.compile('.*
        ResultCardItineraryPlacestyled__StyledResultCardItineraryPlace.*')
13
14      flights = []
15      departure_dates = []
16      departure_info = []
17      return_dates = []
18      return_info = []
19      nights = []
20      prices = []
21
22      for flight in all_flights:
23          nights_spent = flight.find('div', class_='
        ResultCardItinerarystyled__SectorLayoverTextBackground-sc-iwhyue-8
        cYplWP').text.split(' ')[0]
24
25          dates = flight.find_all("p", {"class" : departure_date_regex})
26
27          departure_dates.append(dates[0].find('time').text)
28          return_dates.append(dates[1].find('time').text)
29
30          departure_time = flight.find_all('div',  {"class" :
        departure_info_regex})[0].find('time').text
31          departure_airport = flight.find_all('div',  {"class" :
        departure_info_regex})[0].find('div').text
32          departure_info.append(departure_time + ' ' + departure_airport
        )
33
34
35          return_time = flight.find_all('div',  {"class" :
        departure_info_regex})[2].find('time').text
36          return_airport = flight.find_all('div',  {"class" :
        departure_info_regex})[2].find('div').text
37          return_info.append(return_time + ' ' + return_airport)
38
39          nights.append(nights_spent)
40          prices.append(flight.find('strong',  {"data-test" : '
```

```
    ResultCardPrice'}).text)
41
42
43      df_flights = pd.DataFrame({'Departure date': departure_dates, '
    Departure info': departure_info, 'Return date': return_dates, '
    Return info': return_info, 'Nights' : nights, 'Prices': prices})
44      df_flights.to_csv('flights-belgrade-malaga-results.csv', index=
    False)
```

Оно што треба приметити јесте да се блок са информацијама о одласку са веб-сајта идентификовао коришћењем поклапања по регексу (енг. regex).

Проблем прикупљања података са више страница

TODO: уколико буде превише страница, ова секција се може обрисати јер се пример морао радити коришћењем другог сајта.

У претходно наведеном примеру су вађени подаци из HTML једне веб-странице. Такође је могуће динамично преузети податке и то ће се приказати на примеру веб-страница које не зависе од Јаваскрипта, коришћењем BeautifulSoap и requests пакета. У питању је веб-страница која садржи транскрипте филмова. Да би се успешно извршило преузимање података кроз више страница, треба да се разуме на који је начин то постигнуто на конкретном веб-сајту.

На веб-адреси `https://subslikescript.com/movies_letter-A` уколико се изврши навигација на следећу страницу, може да се примети да се додаје и мења параметар *страница* (енг. page) у веб-адреси и онда она изгледа овако `https://subslikescript.com/movies_letter-A?page=2`. Наведени код ниже са претходно стеченим знањем, проналази блок са нумерацијом страница, извлачи укупан број страница, пролази кроз сваку од њих, поставља вредност изворног HTML на нову тако што креира нову веб-адресу и уз помоћ requests пакета извлачи HTML нове странице, креира нови *BeautifulSoap* објекат за парсирање, проналази све линкове ка филмовима на датој страници, прелази на конкретан линк и преузима транскрипт филма. Кључни моменат јесте да се сваки пут иницијализује *BeautifulSoap* објекат за парсирање.

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 root = 'https://subslikescript.com'
```

```
website = f'{root}/movies_letter-A'
result = requests.get(website)
content = result.text
soup = BeautifulSoup(content, 'lxml')

# pagination
pagination = soup.find('ul', class_='pagination')
pages = pagination.find_all('li', class_='page-item')
last_page = pages[-2].text

links = []

for page in range(1, int(last_page) + 1):
    result = requests.get(f'{website}?page={page}')
    content = result.text
    soup = BeautifulSoup(content, 'lxml')

    box = soup.find('article', class_='main-article')

    for link in box.find_all('a', href=True):
        links.append(link['href'])

    for link in links:
        try:
            result = requests.get(f'{root}/{link}')
            content = result.text
            soup = BeautifulSoup(content, 'lxml')
            box = soup.find('article', class_='main-article')

            title = box.find('h1').get_text()
            transcript = box.find('div', class_='full-script').
    get_text(strip=True, separator=' ')

            print(link)
            with open(f'movies/{title}.txt', 'w') as file:
                file.write(transcript)

        except:
            print('Link not working')
```

3.2 Библиотека Selenium

TODO: Инсталација, сајтови покренути са JS, драјвери, лоцирање елемената, пагинација, имплицитно и експлицитно чекање, попуњавање формулара, логин, кретање између страница, скрејповање странице са бесконачним скролом, како променити корисничког агента и зашто

3.3 Библиотека Scrapy

TODO: Инсталација, scrapy шаблони, креирање паука, стругање са више линкова, стругање са више страница, стругање АПИ, попуњавање формулара, логин, како променити корисничког агента, најосновније потребне функције LUA програмског језика (неопходно за SPLASH), SPLASH, шта представља SPLASH, зашто је неопходан, како превазићи Captcha)

Глава 4

Анализа резултата

TODO: Поређење перформанси, дијаграм односа времена стругања сајта и коришћене библиотеке, како оценити добијене резултате, табела са поређеним карактеристикама коришћених библиотека, препоруке, смернице за унапређење коришћених технологија

Глава 5

Закључак

TODO:

Библиографија

[1] Keio) 1999 W3C® (MIT, INRIA. XPath. on-line at: https://www.w3.org/TR/1999/REC-xpath-19991116/.

[2] Python Software Foundation 2001-2023. html.parser — Simple HTML and XHTML parser. on-line at: https://docs.python.org/3/library/html.parser.html.

[3] Stefan Behnel and Martijn Faassen. Parsing XML and HTML with lxml. on-line at: https://lxml.de/parsing.html.

[4] Osmar Castrillo-Fernández. Web scraping: Applications and tools.

[5] 2023 Software Freedom Conservancy. Selenium. on-line at: https://www.selenium.dev/documentation/.

[6] Gitiles. Headless Chromium. on-line at: https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md.

[7] Ryan Mitchell. Web scraping with python. In *Web Scraping with Python*, 2015.

[8] A Kenneth Reitz P. Requests: HTTP for Humans. on-line at: https://requests.readthedocs.io/en/latest/.

[9] Leonard Richardson. Beautiful Soup Documentation, 2004-2023. on-line at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

Биографија аутора

Зорана Гајић, рођена 04.11.1997 у Москви, где је ... TODO: