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

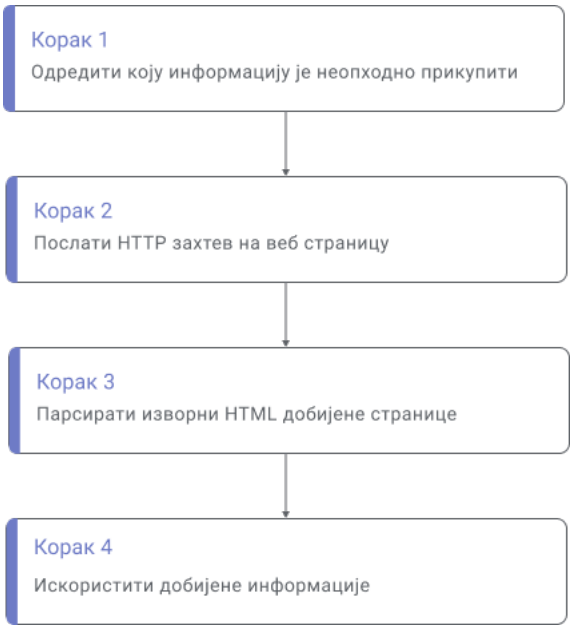
Веб (енг. *World Wide Web, WWW* [1]) представља највећи извор података у историји човечанства, али се већина ових података састоји од неструктурираних информација, што може отежати њихово прикупљање [4]. На многим веб-сајтовима забрањено је копирање и преузимање података, док ручно копирање може потрајати данима или недељама.

Веб скрејпинг представља аутоматизовани процес који омогућава издвајање података са различитих веб-страница и њихово чување у структурираном формату ради тренутне употребе или касније анализе. Овај процес се може написати на различитим програмским језицима, од којих су најпопуларнији Пајтон (енг. *Python*), Јава (енг. *Java*) и Руби (енг. *Ruby*).

Поступак прикупљања информација састоји се од неколико фаза, које су приказане на схеми 2.1. Прва фаза је проналажење одговарајуће веб-странице за прикупљање података (детаљније објашњено у одељку 2.1) и одређивање информација које су потребне за прикупљање. Након тога, потребно је послати *HTTP* (енг. *Hypertext Transfer Protocol* [2]) захтев на жељену веб-страницу и преузети изворни код *HTML* странице. Пре него што се парсира *HTML* код, потребно је пронаћи најбољи начин за индексирање жељених елемената, а затим парсирати изворни код *HTML* странице и извршити неопходну радњу са добијеним информацијама.

[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: Фазе прикупљања података

2.1 Изазови

Веб скрејпинг се сматра корисним алатом за добијање увида у податке. Међутим потребно је пазити на правне аспекте, како би се избегли легални проблеми, јер ниједан веб-сајт не жели да дозволи крађу података. Иако неки веб-сајтови немају механизам одбране, други користе анти-скрејпинг мере, које могу да доведу до блокирања прикупљања података.

Квалитет добијених података је од изузетног значаја за успешно прикупљање информација. Стога треба избегавати веб-сајтове са превише неисправних линкова, јер се веб скрејпинг обично изводи преко целог сајта, а не само преко одређених страница. Када се ради о пројектима великих размера и обимних база података, један од честих изазова јесте складиштење података. Овај проблем може бити решен употребом већ постојећих платформи за складиштење. Када се ради о динамичким веб-сајтовима, решење може бити приказивање у претраживачу без заглавља (енг. *Headless Chromium* [6]), што омогућава да се претраживач покреће у окружењу сервера. Пријављивање корисника на веб-страницу може представљати велики изазов, али се то може решити уз помоћ библиотека као што су *Selenium* и *Scrapy*.

У наставку ће бити описане најчешће праксе које се користе за прикупљање података, а које ће бити укључене у имплементацију описане у секцији

3.2:

1. Уважавање *robots.txt* фајла.

   robots.txt фајл представља политику веб-сајта и најчешће се проналази на нивоу основног директоријума. 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. Промена 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 се користи исти кориснички агент у сваком захтеву, веб-сајт може лако да открије да је у питању аутоматизовани приступ страници.

3. Одговорно коришћење прикупљених података.

   Потребно је да особа која прикупља податке преузме одговорност за њих и пажљиво прати сврху њихове употребе, уз поштовање закона о ауторским правима.

4. Свођење на минимум учесталости прикупљања података.

   Да би се спречило преузимање садржаја са веб-странице, веб-сајт може увести ограничење фреквенције за ботове. Међутим, да би се избегло ово ограничење, обично је довољно омогућити кашњење од 10 секунди између два *HTTP* захтева.

5. Избегавање коришћења увек истог обрасца за прикупљање података. Уместо тога, треба додати радње које симулирају понашање човека, као што је померање миша или клик на насумични линк, како би се остварио природнији утисак.

6. Најбоље време за прикупљање података са веб-сајта јесте када је најмањи број посетилаца активно. Очекује се да ће резултати бити добијени знатно брже током овог периода, него током периода највеће посете.

7. Избегавање прикупљања већ прикупљених података.

8. Промена *IP* адресе (енгл. *Internet Protocol address*) и прокси услуга (енг. *Proxy Services*).

   Када је у питању промена *IP* адресе и коришћење прокси услуга, може се искористи ротирајући прокси сервер. Овај сервер свакој конекцији додељује нову *IP* адресу из скупа доступних проксија.

2.2 Примењене технологије

Веб скрејпинг технологије подразумевају различите методе и алате за издвајање података са веб страница. Регуларни изрази (енг. *Regular Expressions, RegEx*), тагови (енг. *tags*), *CSS* (енг. *Cascading Style Sheets*) селектори и *XPath* (енг. *XML Path Language* [1]) су често коришћене технологије. У наставку ће бити детаљно описано како се свака од ових техника користи у процесу сакупљања података са веб страница. Сви ови алати су од изузетног значаја за успешно издвајање и обраду података са веб страница, и као такви су неопходни за било који веб скрејпинг пројекат.

Регуларни изрази

Регуларни изрази представљају метод за усклађивање специфичних образаца у зависности од датих комбинација, који се могу користити као филтери за добијање жељеног резултата. У прикупљању података регуларни изрази се често користе за поређење шаблона и издвајање података, за локализовање и издвајање специфичних података из *HTML* или *XML* докумената. Једна од најзначајнијих предности регуларних израза јесте у њиховој универзалности, тј. могу се применити на било коју врсту података. У многим програмским језицима, регуларни изрази се подржавају кроз уграђене библиотеке или модуле. На пример, *re* модул програмског језика Пајтон пружа подршку регуларним изразима за поређење шаблона и издвајање података.

Тагови и *CSS* селектори

Тагови

Тагови играју кључну улогу у прикупљању података са веб-страница јер помажу у идентификацији и издвајању одређених информација из изворног кода *HTML* страница. Тагови у *HTML* коду се користе за дефинисање структуре веб странице. Сваки таг представља одређени елемент или секцију странице, као што су заглавља, пасуси, слике и линкови.

У наставку су наведене ознаке које се често користе:

- `<html>`

  Означава почетак и крај *HTML* документа.

- `<body>`

  Представља садржај документа који је видљив кориснику.

- `<h1> до <h6>`

  Користе се за дефинисање наслова.

- `<p>`

  Користи се за дефинисање параграфа текста.

- `<a>`

  Ствара хиперлинк (енгл. *Hyperlink*) до друге веб-странице или ресурса.

- `<ul> и <li>`

  Користе се за стварање неуређене листе ставки.

- `<ol> и <li>`

  Користе се за стварање уређене листе ставки.

- `<div>`

  Користи се за дефинисање одељка документа у сврху стилизовања.

- `<span>`

  Користи се за дефинисање малог дела текста у сврху стилизовања.

- `<input>`

  Користи се за стварање поља за унос корисничких података.

Препоруке за идентификацију елемената у изворном коду *HTML* страница су следеће:

1. Издвајање по идентификатору елемента
2. Издвајање по имену класе елемента
3. Издвајање користећи тагове и *CSS* селекторе
4. Издвајање користећи *XPath*

CSS селектори

CSS селектори се могу користити у процесу сакупљања података са веб-страница како би се идентификовали и издвојили одређени елементи. Коришћењем *CSS* селектора, може се прецизирати које елементе треба издвојити, на основу њиховог имена ознаке, имена класе или идентификатора.

XPath

XPath представља флексибилан начин адресирања различитих делова *XML* (енг. *Extensible Markup Language* [3]) документа. То га чини погодним за навигацију кроз објектни модел документа (енг. *Document Object Model, DOM* [4]) било ког документа сличног *XML*, користећи *XPath* израз (енг. *XPathExpression* [5]). *XPath* израз дефинише образац за одабир скупа чворова и садржи преко 200 уграђених функција. Овај језик је дефинисан од стране *WWW* конзорцијума. У овом раду ће се *XPath* користити за одабир елемената са изворног кода *HTML* страница.

XPath синтакса

XPath користи изразе путања за избор чворова у *XML* документу. Чвор се одабира праћењем путање или корака. Неки корисни примери израза путања су наведени у наставку:

[3] *XML* представља проширив (мета) језик за означавање (енгл. *markup*)
[4] Објектни модел документа представља хијерархијски приказ структуре веб-сајта.
[5] *XPath* израз дефинише образац за одабир скупа чворова.

- `//h2`

  Издваја све *h2* елементе.

- `//div//p`

  Издваја све *p* елементе који се налазе унутар *div* блока.

- `//ul/li/a`

  Издваја све линкове који се налазе унутар неуређених листи.

- `//ol/li[2]`

  Издваја други елемент уређене листе.

- `//div/*`

  Издваја све елементе који се налазе унутар *div* блокова који нису уређени.

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

  Издваја елемент са одређеним идентификатором.

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

  Издваја све елементе са одређеном класом.

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

  Издваја све линкове који имају атрибут *name* или *href*.

- `//a[last()]`

  Издваја последњи линк.

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

  Издваја број редова у табели.

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

  Врши конверзију типова.

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

  Примењује функције са нискама.

- `//*`

  Издваја све елементе.

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

  Издваја све *h1* елементе који се налазе унутар *section* елемента са одређеним идентификатором.

- `//a/text()`

  Издваја текст линка.

- `./a`

  Тачка издваја тренутни чвор.

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

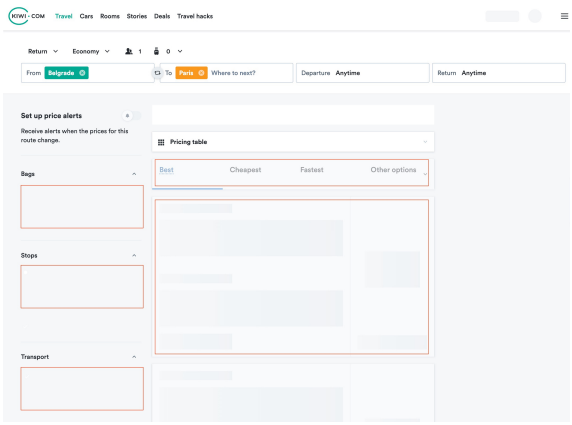
Неопходно је инсталирати *pip3* [1] пакет менаџер и редом *BeautifulSoap*, *requests* [8] и парсер библиотеку *lxml* [3] уз помоћ наведених испод наредби.

```
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

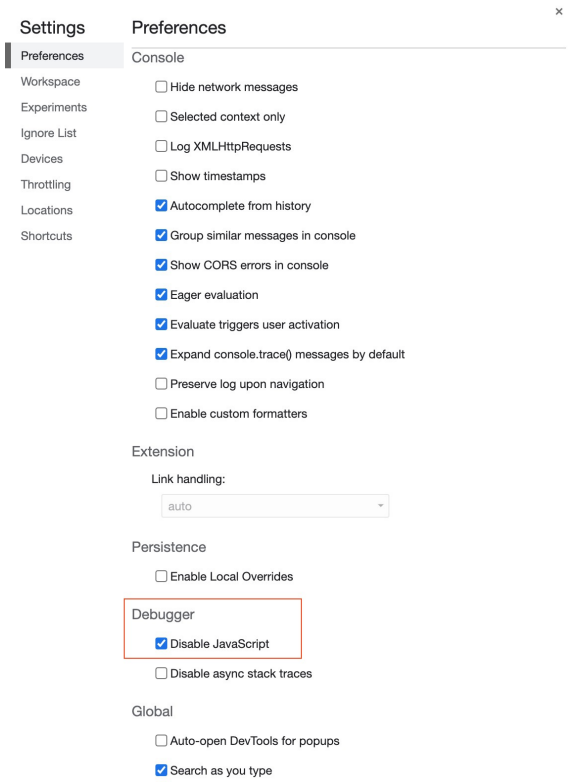
Прва препрека коришћења BeautifulSoap пакета на примеру Киви веб-странице јесте та, да `https://www.kiwi.com/` представља динамичну страницу и да се без Јаваскрипта не може учитати. Самим тим не постоји могућност добијања података конкретно о летовима коришћењем искључиво BeautifulSoap пакета. Изглед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

```
import requests
from bs4 import BeautifulSoup

website = 'https://www.kiwi.com/en/search/results/belgrade-serbia/
    paris-france'
try:
    response = requests.get(website)
except requests.exceptions.RequestException as err:
    print("Error fetching page")
    exit()

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...

```
soup = BeautifulSoup(content, 'lxml')

print(soup.title)
# <title>Belgrade-Paris trips</title>

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

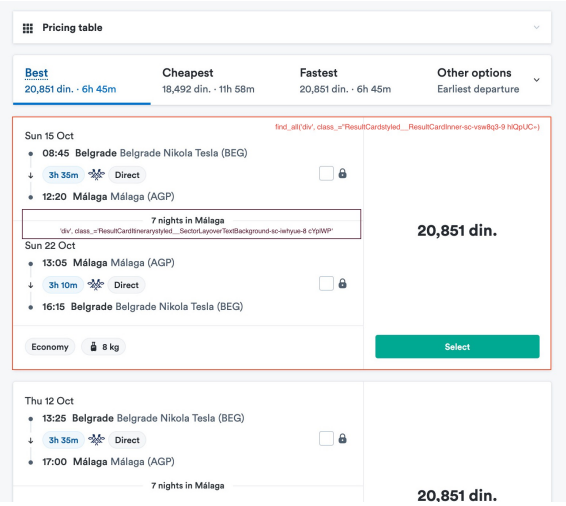
да је сваки лет издвојен у *div* [3] са дугачким именом класе што представља једино по чему се тај блок може идентификовати. Пошто на страници има више летова, најпре се користи функција која проналази све инстанце по потпуној подударности имена класе. На исти начин се извршио проналазак броја ноћи касније.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