Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

Luka Dragar, Matej Vatovec, Aleksander Kovač

EXTRACTION

2. domača naloga za predmet Iskanje in ekstrakcija podatkov s spleta

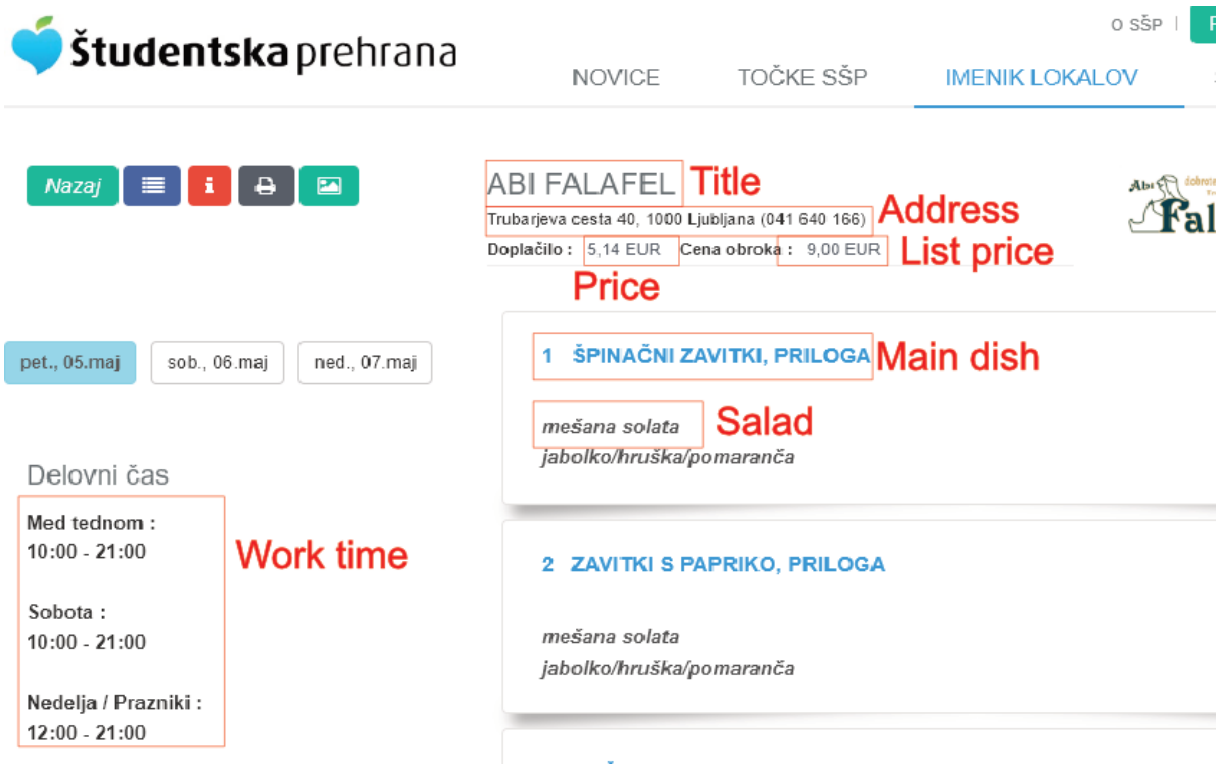
MENTOR: prof. dr. Marko Bajec, doc. dr. Slavko Žitnik

1. Uvod

V poročilu smo povzeli tri načine, kako se lahko izluščijo podatki iz treh različnih spletnih strani. Prva je bila rtvslo.si ter druga overstock.com.

2. Opis izbranih spletnih strani

Za 2 izbirni strani smo si izbrali ponudnike študentske prehrane, kjer smo izbrali katere podatke želimo izluščiti s pomočjo XPATH-a, regexov in algoritma »Road Runner«. Sledeča slika prikazuje izbrane parametre.



.

3. Implementacija

Ukaz »pip install -r requirements.txt namesti potrebne pakete potem pa zaženemo run-extraction.py skripto ter odvisno od zahtevanega algoritma dodamo arugment (A – regex, B – xpath, C – »Road Runner«).

3.1 Tabela regularnih izrazov in XPath za posamezne strani

REGEX

RTV Slo″

```
regex_list = [
    (r"<h1>(.*?)</h1>", "Title"),
    (r"<div class=\"subtitle\">(.*?)</div>", "SubTitle"),
    (r"<p class=\"lead\">(.*?)</p>", "Lead"),
    (r"<div class=\"author-name\">(.*?)</div>", "Author"),
    (r"<div class=\"publish-meta\">\n\t\t(.*?)<br>", "PublishedTime"),
    (r"<div class=\"article-body\">(.*?)<div class=\"gallery\">", "Content")
]
```

Overstock

```
regex_list = [
(r"<td><a href=\"http://www\.overstock\.com/cgi-bin/d2\.cgi\?PAGE=PROFRAME[\w\W]*?\"><b>(.*?)</b>", "Title"),
(r"<s>(.*?)</s>", "List price"),
(r"<span class=\"bigred\"><b>(.*?)</b>", "Price"),
(r"<b>You Save:[\w\W]*?class=\"littleorange\">(.*?)</span>", "Saving"),
(r"<span class=\"normal\">(.*?)</span>", "Content")
]
```

Študentska prehrana

```
regex_list = [
      (r"<h3 class=\"no-margin bold\">(.*?)</h3>", "Title"),
      (r"<small>(.*?)</small>[\w\W]*", "Address"),
      (r"<span class=\" color-light-grey\">(.*?)</span>", "Price"),
      (r"<span class=\" color-light-grey\">(.*?)<\/span>\s*<\/small>", "List price"),
      (r"<div class=\"col-md-12 text-bold\">(.*?)</div>", "Work time"),
      (r"<strong class=\" color-blue\">(.*?)</strong>", "Main dish"),
      (r"<ul class=\"list-unstyled\">(?:\s*<li>.*?</li>){1}\s*<li>.*?<i>(.*?)</i>", "Sa-
lad"),
      ]
```

XPATH

RTV Slo

```
for key, xpath in {
    "Title": '//header[@class="article-header"]/h1',
    "SubTitle": '//div[@class="subtitle"]',
                "Lead": '//p[@class="lead"]',
                "Author": '//div[@class="author"]/div',
                "PublishedTime": '//div[@class="publish-meta"]/text()[1]',
                "Content": '//article[@class="article"]', #regex would be a better option for this case.
}.items():
    element = site_string.xpath(xpath)
    if element:
```

Overstock

```
common_path = f'{common_path}table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr[1]/td[2]'
data = {}
for key, xpath in {
    "Title": f'{common_path}/a/b',
    "List price": f'{common_path}/table/tbody/tr/td[1]/table/tbody/tr[1]/td[2]/s',
    "Price": f'/{common_path}/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/span/b',
    "Saving": f'{common_path}/table/tbody/tr/td[1]/table/tbody/tr[3]/td[2]/span',
    "Content": '/html/body/table[2]/tbody/tr[1]/td[5]/table/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr[3]/td[2]/table/tbody/tr/td[2]/span'
```

Študentska prehrana

```
data['Locale name'] = single_value(
    site_string, f'{common_path}/div[1]/div[1]/div[2]/div[1]/h3[1]')
data['Address'] = single_value(
    site_string, f'{common_path}/div[1]/div[1]/div[2]/div[1]/small[1]')
data['Price'] = single_value(
    site_string, f'{common_path}/div[1]/div[1]/div[2]/div[1]/small[2]/span[2]')
data['List price'] = single_value(
    site_string, f'{common_path}/div[1]/div[1]/div[2]/div[1]/small[2]/span[4]')
data['Work time'] = single_value(
    site_string, f'{common_path}/div[2]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[1]/div[2]/div[1]')  # need combining with regex to arhieve best result
data['Salad'] = single_value(
    site_string, '//ul[@class="list-unstyled"][1]/li[2]/i[1]')
data['Main dish'] = single_value(
    site_string, f'//strong[@class=" color-blue"][1]')
print(json.dumps(data, indent=4, ensure_ascii=False))
```

3.2 Road runner

Ideja naše implementacije zaznavanja poljubnih elementov in iteratorjev z algoritmom Road runner je sledeča.

Kot začetni wrapper vzamemo prvo stran, ki jo lahko poljubno predprocesiramo. Nekatere dele strani lahko že takoj spremenimo v regex izraze.

Recimo <script> ali <style> značke. Tako imamo wrapper, ki je sestavljen iz značk, nizov in regex izrazov.

Nato začnemo primerjati wrapper z naslednjo stranjo. Po vrsti vzemamo elemente iz wrapperja in gledamo če jih najdemo na vrhu druge strani. Če dobimo ujemanje, posodobimo wrapper in odstranimo element iz druge strani. Če ne preverimo ali smo naleteli na iterator ali na poljubni element.

Podani sta psevdokodi za detekcijo iteratorja in detekcijo poljubnih elementov.

Psevdokoda za detekcijo iteratorja:

```
function detect_iterators(wrapper, sample, missmatch):
    output: wrapper, if wrapper was changed else None
    let terminal_tag = wrapper[missmatch-1] be the closing tag
    B1= []
    for line, index in wrapper[missmatch:]:
        if tag(line) == terminal_tag:
            B1= wrapper[missmatch:index]
            break
        else number of closing tabs > number of openning tabs:
            return None
    for line in B1:
        if line does not match wrapper upwards:
            changes = detect_iterators()
            if iterator is None:
                changes = detect_optionals()
            update_wrapper(changes)
    repeat for sample
    return wrapper
```

Iskanje poljubnega elementa:

```
for wrapper_element in wrapper_elements:
    if is_match(wrapper_element, second_page):
        new_wrapper += wrapper_element
    else:
        is_found, second_page_element_location =
            search_for_element(wrapper_element, second_page_element)
        if is_found:
            optional_element =
                second_page[:second_page_element_location]
            second_page = second_page[second_page_element_location:]
            new_wrapper += optional_element
        else:
            second_page = old_second_page
            is_found, wrapper_element_location =
                search_for_element(second_page_element, wrapper_elements)
            if is_found:
                optional_element =
                    wrapper_elements[:wrapper_element_location]
                wrapper_elements =
                    wrapper_elements[wrapper_element_location:]
                new_wrapper += optional_element
            else
                new_wrapper += optional_elements_regex
```

Tako ponovno dobimo wrapper sestavljen iz značk, nizov in regex izrazov, ki ga lahko apliciramo na poljubno število nadaljnih strani.