Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/03BE09Frank

JEZIK ZA PODRŠKU EKSTRAKCIJI PODATAKA IZ NESTRUKTURIRANIH IZVORA SA VEBA

A LANGUAGE FOR UNSTRUCTURED WEB-BASED DATA EXTRACTION

Filip Frank, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisan je DSL (eng. Domain Specific Language) za ekstraktovanje podataka sa veba, napisan uz pomoć textX biblioteke i Python programskog jezika..*

Ključne reči: *DSL, struganje, Scrapy, textX, ekstrakcija, Python*

Abstract – *In this paper, DSL for extracting data from Web resources is described, created using textX library and Python programming language.*

Keywords: *DSL, scraping, Scrapy, textX, extraction, Python*

1. UVOD

Osnovna ideja pisanja jezika za podršku ekstrakciji podataka jeste implementirati jasan i koncizan jezik, ograničen na domen veb ekstraktovanja, kojim će se svlačiti određene informacije definisane upitima. Jezik se pritom nece ograničavati na jedan izvor podataka (jednu veb stranicu) već će omogućavati prilagođavanje različitim domenima i strukturama stranice.
Cilj je da programer ima potpunu slobodu u definisanju modela podataka i formata na koji će se ti podaci čuvati. U objektno-orjentisanom svetu, svaki entitet se može opisati odgovarajućom klasom i atributima, pa je bitno identifikovati entitete značajne za proces ekstraktovanja, uzimajući u obzir informacije dostupne u izvoru podataka. Na taj način, definisanje modela će se zasnivati na identifikaciji značajnih elemenata strukture izvora podataka sa stanovišta informacija koje sadrže i njihovo prevođenje na model.

2. TEORIJSKE OSNOVE

2.1 World Wide Web

World Wide Web, iliti samo Veb, predstavlja informacioni prostor u kojem se dokumenti i drugi veb izvori identifikuju jedinstvenim lokatorima (URL - *Uniform Resource Locators*) i gde su ti dokumenti povezani hipertekstualnim vezama i dostupni preko Interneta [1].
Sa stanovišta ekstraktovanja i svakodnevne upotrebe Interneta, korišćenjem veb pregledača, komunikacija se odvija korišćenjem HTTP protokola.

HTTP je protokol za transport HTML dokumenata između servera i klijenta. Najbitnije karakteristike ovog protokola jeste zahtev/odgovor komunikacija. HTTP je u suštini jednostavan mrežni protokol, baziran na tekstualnim dokumentima (porukama) propisanim po HTTP standardu.

HTML (*Hyper Text Markup Language*) je jezik koji opisuje format i strukturu veb stranica. Iz ugla HTTP komunikacije, HTML opisuje sadržaj tela HTTP odgovora poslatog od strane HTTP servera. Pošto se upravo u HTML-u nalaze podaci koji su od važnosti za ekstraktovanje, ovaj elemenat je ključan u Internet komunikaciji.
HTML je „*markup*" jezik, iz razloga što tag može da definiše i strukturu i način na koji će se sadržaj koji tag ima prikazati korisniku.

CSS (*Cascade Style Sheet*) je jezik za definisanje izgleda veb stranice. Razlika u odnosu na HTML jeste, pre svega, sintaksa. Definiše se kao lista izraza razdvojenih tačka-zarezom pri čemu se izrazi navode u obliku *naziv:vrednost*.
CSS selektori su bitni pošto služe za selektovanje odgovarajućih elemenata koristeći CSS osobine HTML elementa.

2.2 Struganje

Termin struganja označava proces automatskog prikupljanja podataka sa Interneta (eng. *scraping*).
Sa stanovišta struganja, API servisi su najbolji izvor informacija. Podaci koje on pruža su potpuni, ažurni, koncizni, standardizovani, u unapred definisanom formatu i dokumentovani, dostupni preko URL-a.

2.3 Struganje veb stranica

Veb stranice su namenjene krajnjem korisniku stoga sadrže najviše informacija koje su nam potrebne, one su ažurne i lako im je pristupiti. Nema dokumentacije niti posebne provere da li su podaci ažurirani i glavni zadatak je pronaći i ekstraktovati elemente koji su nosioci traženih informacija. Svaki sadržaj, koji je prikazan na stranici, može biti ekstraktovan [2].

2.4 Puzanje

Puzanje (eng. *crawl* – puzanje, mileti) predstavlja proces u kom se struganje ne ograničava na jednu stranicu već softver „puzi'' kroz više stranica čije su adrese ekstraktovane parsiranjem stranice. U teoriji, puzanje je

---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Igor Dejanović, vanr. prof.

termin koji označava sposobnost programa da automatski ekstraktuje stranice i da sam istražuje određeni domen [3]. Dobra praksa je razvdojiti procese puzanja i struganja, tako da je logika pronalaženja novih stranica razdvojena od logike parsiranja stranice.

Nakon uspešnog pronalaska podataka koji se ekstraktuju, potrebno je sačuvati te podatke u nekom obliku. Dobar proces struganja se ogleda u uspešno prikupljenim podacima koji su dostupni za dalje analize nakon što se čitav proces završi. Način na koji će se ti podaci čuvati zavisi dosta od tipa podataka koji se sakuplja i njihove svrhe.

2.4 Jezici specifični za domen

Jezici opšte namene (JON) obuhvataju široku oblast problematike jer je osnovna ideja prilikom njihovog kreiranja upravo sposobnost rad sa problemima iz različitih oblasti.
Za razliku od njih, postoje jezici koji pokrivaju samo usko ograničen domen i čija namena je da se prilagode problematici tog domena u cilju veće efikasnosti i produktivnosti u odnosu na jezike opšte namene. Ovakve jezike nazivamo jezicima specifičnim za domen – JSD (DSL – *Domain Specific Language*) [4].
Glavne karakteristike JSD jezika:

- prilagođenost domenu problema,
- visok nivo apstrakcije,
- konciznost.

3. PREGLED I KORIŠĆENE TEHNOLOGIJE

3.1 Korisni moduli

Python sadrži mnoštvo modula koji podržavaju ovakav tip komunikacije: *urllib*, *urllib3*, *httplib2*, *requests*, *aiohttp*, itd [3]. *Requests* modul u potpuno podržava rad sa HTTP komunikacijom, slanje HTTP zahteva, modifikaciju zaglavlja, rad sa formama, slanje kolačića itd.

BeautifulSoup je *Python* biblioteka za parsiranje HTML i XML dokumenata. Pruža splet raznoraznih funkcija za pronalazak elemenata, navigaciju i modifikovanje parsiranog stabla.

Treba spomenuti i jednu od alternativa BeautifulSoup-u, *lxml* modul. Glavne karaktestike ovog modula su robustnost, brzina, podrška za *Xpath* selektore. Brzina izvršavanja i veliki broj funkcija je glavna prednost u odnosu na druge alate. Osnovna ideja je ista, programski se kreira stablo na osnovu unetog HTML koda a zatim se pozivaju funkcije za pronalazak I manipulaciju elemenata stabla. Za razliku od *BeautifulSoup*-a, *lxml* podržava mehanizam i CSS i *Xpath* selektora.

Selenium je alat koji služi prvenstveno za testiranje veb stranica ali je između ostalog i snažan alat za ekstraktovanje. Integriše se u postojeće veb pregledače i automatizuje ih u cilju testiranja i prikupljanja podataka ili izvršavanja bilo kakve akcije na stranici. Podržan je od raznih programskih jezika, između ostalog i u Pajtonu.

3.1 Scrapy

Scrapy je softversko okruženje, napisano u Pajtonu, za puzenje po veb sajtovima i ekstraktovanje podataka koji se mogu koristiti za razne aplikacije, prikupljanje podataka, procesiranje informacija i ostalo [5]. Objedinjuje sve funkcionalnosti potrebne za struanje: ekstraktovanje novih linkova, puzanje po čitavim domenima i pronalaženje elemenata u parsiranim stranicama. *Scrapy* koristi termin pauk (eng. *spider*) za klase koje prikupljaju informacije sa veb stranice.

3.2 Arpeggio

Arpeggio je parser, napisan u *Python* jeziku i njegova osnovna namena je podrška za pisanje JSD jezika [6]. Glavne karakteristike ovog parsera jeste da je on rekurzivni silazni parser sa vraćanjem i memorizacijom. Omogućava definisanje gramatike, parsiranje na osnovu definisinanih gramatičkih pravila i interpretaciju koda.

3.2 textX

TextX je JSD jezik za definisanje drugih JSD jezika koristeći *Python*. Na osnovu definisane gramatike, *textX* kreira meta-model i parser za definisani JSD jezik [7]. Gramatika se zasniva na *textX* pravilima koja se definišu u *.tx* fajlovima. Osnovna struktura *textX* pravila je naziv pravila (parser prihvata slova, cifre i donje crte), dvotačka, *textX* izraz i na kraju tačka-zarez. U okviru *textX* izraza koristimo ugrađene tipove, druga definisana pravila i *textX* operatore.

3.3 Jinja2

Jinja2 je obrađivač šablona namenjen za *Python* programski jezik [8]. Zgodan je za generisanje bilo kakvog programskog koda, i sličan je *Django* šablonima. Najveće prednosti korišćenja *Jinja2* obrađivača je: automatsko eskejpovanje HTML-a korišćenjem specijalnih karaktera, mehanizam nasleđivanja šablona, sloboda u konfigurisanju sintakse, *sandbox* izvršavanje koje pruža zaštićeno okruženje za automatizovano testiranje programa.

3.4 MongoDB

Za persistenciju podataka sakupljenih struganjem, u implementiranom rešenju koristimo *MongoDB*. *MongoDB* je *open-source* baza podataka u kojoj se podaci čuvaju u obliku dokumenata. Glavne karakteristike su dobre performanse, dostupnost i automatsko skaliranje.

4. IMPLEMENTACIJA SISTEMA

4.1 Definisanje gramatike

Kreirani jezik za definisanje svih izraza i upita u implementiranom rešenju zasniva se na gramatikama napisanim korišćenjem *TextX* alata, tačnije na jednoj gramatici koja definiše konfiguraciju i deklarisanje modela podataka (tipova) i drugoj, koja definiše pisanje jezičkih upita.

Gramatika za deklarisanje entiteta koji se ekstraktuju treba da podrži sledeće funkcionalnosti:

- Definisanje jednog ili više tipova entita.
- Za svaki entitet, definisanje njegovog imena i jednog ili više atributa.
- Za svaki atribut, definisanje CSS selektora koji jedinstveno identifikuju elemenat na željenoj stranici.
- Ukoliko su traženi podaci u okviru tabele ili (ne)uređene liste na veb stranci, mogućnost definisanja takvih elemenata i CSS selektora koji identifikuju ove elemente. Definisati CSS selektor elementa koji će iterirati kroz pomenute strukture.

Koristeći napisanu gramatiku, jezikom je pokriveno definisanje tipova podataka, njihovo povezivanje sa elementima nestrukturiranih izvora (HTML elemenata veb stranice) i navođenje elemenata koji su nosioci više parametara definisanih tipova podataka.

```
Movie (
title {"div.title_wrapper h1::text"},
year {"div.title_wrapper h1 span#titleYear a::text"},
rating {"div.ratingValue strong span::text"},
duration{"div.subtext time::text"},
genre{"div.subtext a::text"},
release_date {"div.subtext a::text"},
story_line{"div.plot_summary div.summary_text::text"},
director {"div.credit_summary_item a::text"},
writers{"div.credit_summary_item a::text"},
cast {"div.article table.cast_list"}
 )
```

Slika 1. *Primer deklarisanja tipa*

Gramatika za deklarisanje entiteta koji se strugaju treba da podrži sledeće funkcionalnosti:

- Definisanje jednog ili više tipova entita.
- Za svaki entitet, definisanje njegovog imena i jednog ili više atributa.
- Za svaki atribut, definisanje CSS selektora koji jedinstveno identifikuju elemenat na željenoj stranici.
- Ukoliko su traženi podaci u okviru tabele ili (ne)uređene liste na veb stranci, mogućnost definisanja takvih elemenata i CSS selektora koji identifikuju ove elemente. Definisati CSS selektor elementa koji će iterirati kroz pomenute strukture.

```
find Movie where cast = 'Morgan Freeman'
```

Slika 2. *Primer definisanja upita*

4.2 Podrška za rad sa ekstraktovanim sadržajem

Ideja je da koristeći *Python* definišemo funkcije za rad sa sadržajem ekstraktovanih podataka pri čemu nam se pružaju sve pogodnosti korišćenja *Python* jezika: definisana sintaksa, postojeće funkcije za rad sa stringovima, editori, sintaksno bojenje itd. Na ovaj način, definisani JSD jezik će ostati nepromenjen i potrebno je samo integrisati ručno-pisani i generisani kod, odnosno *Python* funkcije vezati za konkretne parametre definisanih tipova. Rezultat izvršavanja svake funkcije biće string koji se dodeljuje vrednosti tog parametara.

```
def Movie_genre(content):
    return ", ".join(content.extract()[:-1])

def Movie_release_date(content):
    return content.extract()[-1].strip()

def Movie_story_line(content):
    return content.extract_first().strip()
```

Slika 3. *Primer Python funkcija*

Pošto gramatika podržava definisanje elemenata koji sadrži više parametara određenog tipa (tabela, uređena ili neuređena lista), *Python* funkcije se koriste prilikom definisanja načina ekstrakcije informacija iz ovakvih struktura.

Mehanizam traženja parametara se zasniva na pronalasku takvih elemenata na veb stranici korišćenjem definisane gramatike. Pisanjem deklaracije tipova i navođenjem CSS selektora koji ih identifikuju, uspešno pronalazimo na stranici ovakve elemente i elemenat kojim prolazimo kroz strukturu i pronalazimo podatke.
Sledeće što je potrebno definisati su načini prepoznavanja značenja tih podataka, odnosno mapiranje parametara i izvučenih podataka, na osnovu vrednosti određenih elemenata na stranici.

```
Table (
        tag {"div.product-panels-content table.data-table"},
        row {"tr"}
)
```

Slika 4. *Primer deklarisanja tabele*

```
def Table_name(row_selector):
    return row_selector.css('th::text').extract_first().strip()

def Table_value(row_selector):
    return row_selector.css('td::text').extract_first()
```

Slika 5. *Primer Python funkcija za rad sa tabelom*

Mehanizam „normalizovanja'' vrednosti izvučenih iz tabela smanjuje osetljivost programa na odstupanja između naziva parametara i naziva kolona/reda na način što briše sve specijalne karaktere, razmake zamenjuje donjom crtom (prilagođavanje pravilima gramatike definisanja naziva parametara) i sve karaktere pretvara u mala slova. Ovakvim normalizovanjem izbegavaju se situacije ne poklapanja dveju vrednosti zbog malih i velikih slova ili razmaka.

Mehanizam pseudonima, podržan od definisane gramatike, omogućava definisanje nekoliko alternativnih naziva za isti parametar i program će, ukoliko ne dođe do poklapanja dva naziva, porediti sve pseudonime za dati parametar i pokušati da nadomesti ovakav problem.

4.3 Šabloni

Nakon parsiranja deklaracije tipova i upita na osnovu definisanih gramatika, dobijamo dva modela, po jedna za oba JSD izraza. Da bi iskoristili ove modele za generisanje koda *Scrapy* projekta, potrebni su nam *Jinja2* šabloni. Imajući u vidu strukturu *Scrapy* projekta, kreirani

su šabloni koji odgovaraju imenima ciljanih fajlova: *items, pipelines, settings, middlewares...*

Šabloni za generisanje pauka će biti jedinstven za svaki domen, za koji radimo struganje. Glavna struktura pauka se svodi na pisanje *start_requests* metode, koja će poslati inicijalni zahtev na specificiranu stranicu, i metode koja će ekstraktovati linkove i slati zahteve ka stranicama koje sadrže podatke o entitetu koji se ekstraktuje. Metode se navode direktno u šablonu, i nepromenjene se generišu u krajnji kod.

```
def start_requests(self):
        url = "https://www.imdb.com/chart/top?ref_=nv_mv_250"
        return [scrapy.Request(url, callback=self.main_page)]

def main_page(self,response):
        links = response.css("table.chart.full-width tbody")
        for link in links.css("tr"):
                yield scrapy.Request("https://www.imdb.com"+
link.css("td.titleColumn a::attr(href)").extract_first())
```

Slika 6. *Primer spider metoda*

Šablon za osnovni oblik *parse* funkcije zavisi od karakterističnosti stranice koja se ekstraktuje. U njemu se instancira objekat klase definisane u *items.py* modulu, zajedno sa svim parametrima, kojima se dodeljuje vrednost na osnovu *response* objekta i CSS selektora unetih na ulazu.

```
def parse(self,response):
                movie = Movie()

                title = response.css("div.title_wrapper h1::text")

                if "Movie_title" in dir(custom_module):
                        movie['title']= getattr(custom_module,
'Movie_title')(title)
                else:
                        movie['title'] = title.extract_first()
```

Slika 7. *Primer dela parse metode*

Način integracije generisanog i ručno pisanog koda, definisanog u propratnom .py modulu, koji se prilikom generisanja koda smešta u isti folder gde i *spider.py* fajl, može se primetiti na datom primeru. Ukoliko postoji funkcija za taj parametar, definisana u *custom_module*, ona će biti pozvana, u suprotnom, prosleđujemo komletan sadržaj ekstraktovanog elementa.

Osnova prilikom generisanja koda za *pipelines.py* je nepromenjiva: kod će validirati podatke na osnovu određenih uslova i ukoliko su oni zadovoljeni, čuvaće ih u *MongoDB* bazu. Uslovi za validiranje su određeni upitom, odnosno kriterijumima upita.
Metoda ___*init*___ će referencirati podešavanja vezana za bazu podatka, definisana u *settings.py*, koja će biti nepromenjiva za sve aplikacije. Metoda *process_item* - sadrži logiku validiranja podataka i u slučaju uspeha, čuvanja podataka u bazu. Šablon će proći kroz sve kriterijume definisane u upitu i smestiti u jedan *if* uslov, tako da za sledeći upit:

U *settings.py* modulu se nalaze sva projektna podešavanja vezana za ponašanje bota, podešavanje zaglavlja, aktiviranje *pipeline* klasa, parametri baze podataka itd.

4.4 Python generator

Namena *Python* generatora je da objedini sve što je navedeno u implementaciji rešenja: upotrebom *textX* da generiše meta-model i model podataka na osnovu definisanih deklaracija tipova i upita, imajući u vidu gramatiku, i da zatim dobijene modele prosledi spomenutim šablonima koristeći *Jinja2* klase.

5. ZAKLJUČAK

Nedostatak kvalitetnog i konciznog JSD-a koji se bavi struganjem, osnovni je razlog implementiranja ovog rešenja. Upoznati sa prednostima struganja i korišćenja jezika specifičnih za domen, ideja je implementirati jasan i precizan jezik za struganje veb stranica. Sa njim ćemo kreirati entitete i upite koji će se lako prilagođavati različitim strukturama, koji će omogućavati filtriranje željenih podataka i koji će brzo i efikasno rezultirati sa podacima sačuvanim u bazi podataka.

Dalji pravci razvoja jezika su pre svega usmereni na kreiranje modela pa zatim i generisanje koda vezanog za definisanje stranice koja se ekstraktuje i način na koji će se linkovi ekstraktovati (CSS selektori u kojima se nalaze linkovi). Trenutno se ova logika nalazi u konkretnim šablonima za željeni domen.
Pored toga, trenutno se process ekstraktovanja zasniva na CSS selektorima, pa bi podrška *Xpath* selektorima bila značajna u situacijama kada nam je potrebna sva moć i fleksibilnost *Xpath* jezika.

6. LITERATURA

[1] Wikipedia, „World Wide Web,“ [Na mreži]. Available: https://en.wikipedia.org/wiki/World_Wide_Web. [Poslednji pristup 10 October 2018].
[2] H. Brody, The Ultimate Guide to Web Scraping, 2017.
[3] S. vanden Broucke i B. Baesens, Practical Web Scraping for Data Science, Apress, Berkeley, CA, 2018.
[4] I. Dejanovic, „Jezici specifični za domen,“ [Na mreži]. Available: http://www.igordejanovic.net/courses/jsd.html. [Poslednji pristup 3. oktobar 2018].
[5] Scrapy, „Scrapy 1.5 documentation,“ [Na mreži]. Available: https://doc.scrapy.org/en/latest. [Poslednji pristup 4. oktobar 2018].
[6] I. Dejanovic, „Arpeggio Documentation,“ [Na mreži]. Available: http://www.igordejanovic.net/Arpeggio. [Poslednji pristup 6. oktobar 2018].
[7] I. Dejanovic, „Arpeggio,“ [Na mreži]. Available: http://www.igordejanovic.net/courses/tech/arpeggio.html. [Poslednji pristup 6. oktobar 2018].
[8] Jinja2, „Jinja2 Documentation,“ [Na mreži]. Available: http://jinja.pocoo.org/docs/2.10/api. [Poslednji pristup 7. oktobar 2018].

Kratka biografija:

Filip Frank rođen je u Novom Sadu 1992. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo, odbranio je 2018.god.