Programming Assignment 2

Implementing structured data extraction

Jaka Kokošar, Danijel Maraž, and Toni Kocjan

Fakulteta za Računalništvo in Informatiko UL dm9929@student.uni-lj.si, jk0902@student.uni-lj.si, tk3152@student.uni-lj.si

Povzetek The article covers the work done in the scope of the second programming assignment as part of the subject web information extraction and retrieval.

Keywords: Data Extraction Retrieval XPath Regex Roadrunner

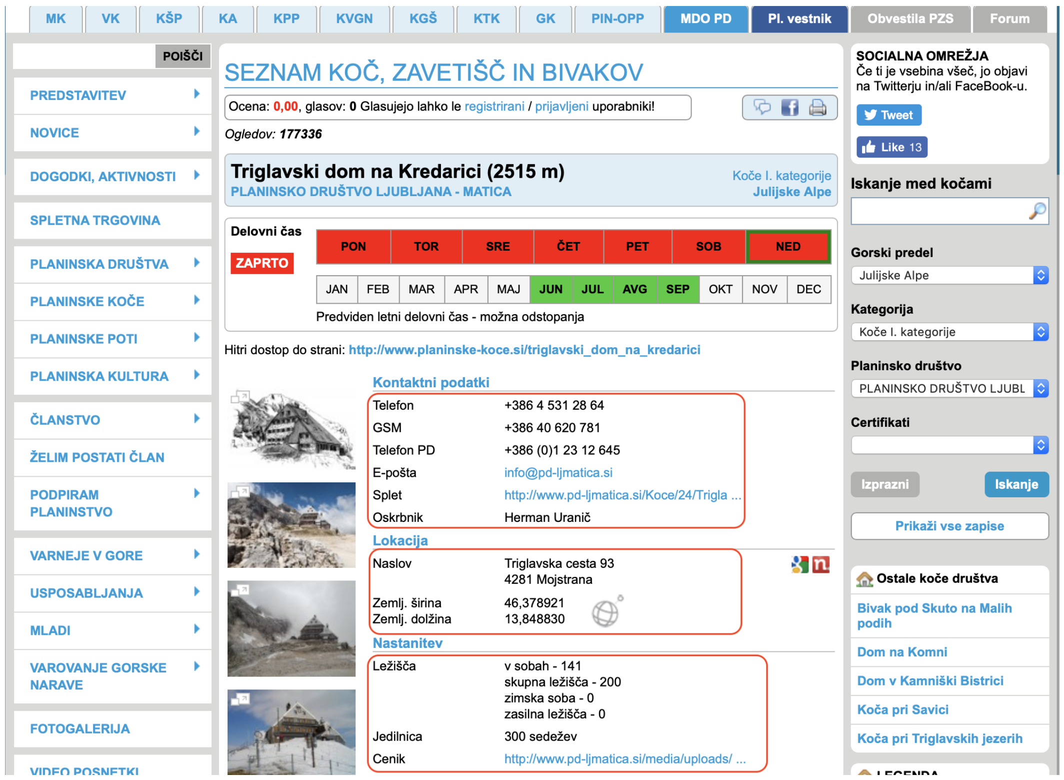
1 Introduction

After having collected the raw data with a crawler the logical next step is to convert it into a more structured format. As web pages do not have a strict shape this poses quite a challenge as any attempt of data extraction must be robust and resistant to various deformities in the html code. The report covers our attempts in implementing processes of structured data extraction in 6 different pages using basic methods such as regular expressions and xpath expressions. Alongside this an attempt was made to implement the RoadRunner algorithm which greatly simplifies the unpredictability aspect of data extraction.

2 Description of chosen web pages and data items

In addition to the mandatory websites, we selected a PZS[1] (*Planinska zveza Slovenije*) site, which is the central register of information about the operation of mountain huts in Slovenia. As shown in Figure 1, we wanted to extract information of accommodations, location and contact details of a given hut. Since there are no other resources or APIs through which we could access this information and use them for any other purpose, we are forced to obtain information directly from this site.

[1] https://www.pzs.si/



SEZNAM KOČ, ZAVETIŠČ IN BIVAKOV

Ocena: 0,00, glasov: 0 Glasujejo lahko le registrirani / prijavljeni uporabniki!

Ogledov: 177336

Triglavski dom na Kredarici (2515 m) Koče I. kategorije
PLANINSKO DRUŠTVO LJUBLJANA - MATICA Julijske Alpe

Delovni čas

ZAPRTO

PON	TOR	SRE	ČET	PET	SOB	NED

JAN	FEB	MAR	APR	MAJ	JUN	JUL	AVG	SEP	OKT	NOV	DEC

Predviden letni delovni čas - možna odstopanja

Hitri dostop do strani: http://www.planinske-koce.si/triglavski_dom_na_kredarici

Kontaktni podatki

Telefon	+386 4 531 28 64
GSM	+386 40 620 781
Telefon PD	+386 (0)1 23 12 645
E-pošta	info@pd-ljmatica.si
Splet	http://www.pd-ljmatica.si/Koce/24/Trigla ...
Oskrbnik	Herman Uranič

Lokacija

Naslov	Triglavska cesta 93 4281 Mojstrana
Zemlj. širina	46,378921
Zemlj. dolžina	13,848830

Nastanitev

Ležišča	v sobah - 141 skupna ležišča - 200 zimska soba - 0 zasilna ležišča - 0
Jedilnica	300 sedežev
Cenik	http://www.pd-ljmatica.si/media/uploads/ ...

SOCIALNA OMREŽJA
Če ti je vsebina všeč, jo objavi na Twitterju in/ali FaceBook-u.

Iskanje med kočami

Gorski predel: Julijske Alpe

Kategorija: Koče I. kategorije

Planinsko društvo: PLANINSKO DRUŠTVO LJUBL

Certifikati

Ostale koče društva

- Bivak pod Skuto na Malih podih
- Dom na Komni
- Dom v Kamniški Bistrici
- Koča pri Savici
- Koča pri Triglavskih jezerih

Slika 1. Data items we want to extract from pzs.si website

3 Regular expressions implementation

We defined three functions *parse_rtv_content*, *parse_overstock_content*, *parse_pzs_content*. Every function is responsible for a different page type but they are structured similarly. We extract data in three steps:

1. read the input file
2. pattern matching
3. clean the results

We only define one regular expression per page type. We extract all of the data items with a single expression in which we define all of the desired groups. After we obtain the results we clean the data of leftover html tags and strip newlines, whitespaces and tabulators. Results are available in the project repository[2].

[2] https://github.com/LampDM/structured_data_extraction_methods/tree/master/output/regex

4 XPath implementation

4.1 jewelry

An initial Xpath expression is called on the input html which selects a *tbody* element. The children of this element represent our jewelry and other items on the page. We count the number of children and with a for loop and extra Xpath expression access each data item that interests us for each child. Any children that are malformed or do not contain the items we're interested in (such as price etc.) encounter an exception as the Xpath expression throws an error and are automatically not processed. Afterwards some minor post-processing of the acquired strings is done and the json is created.

Output

Output

4.2 cars

The items Title, SubTitle, Author, PublishedTime and Lead are all collected via seperate XPath expressions on the input html as both pages have the same structure when it comes to them. The main difference between the two pages is the structure of the article body from which we've derived our Content item. We handle this by first extracting an article body tag which contains all the contents we need. Afterwards we give the extracted element as an argument to our function *intr* which recursively iterates through the tag structure and appends all encountered text into a *nonlocal* string. The string is then briefly processed and added to our json output.

output1

output2

4.3 koce

output1

output2

5 RoadRunner like implementation

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```