# Seminar 2

Edo Ljubijankić

Fakulteta za računalništvo in informatiko, Univerza v Ljubljani,Večna pot 113, Slovenija
el0978@student.uni-lj.si

## 1 Izbira strani

Za ekstrakcijo podatkov smo pri izbiri dveh podobnih strani z isto domeno izbrali avto.net. Stran vsebuje seznam avtomobilov, ki jih uporabnik želi pogledati. Ena stran vsebuje daljši seznam prikazanih avtomobilov, druga manj in smo zaradi seznama procesirali vse podatke s seznama podobno kot za Overstock stran. Izbranih podatkov za ekstrakcijo je 6 in sicer: Ime oglasa, Letnik 1.registracije, Prevoženi km, Motor, Vrsta menjalnika in Cena. Te podatki se nahajo na desni strani slike oglasa (primer na sliki 1).

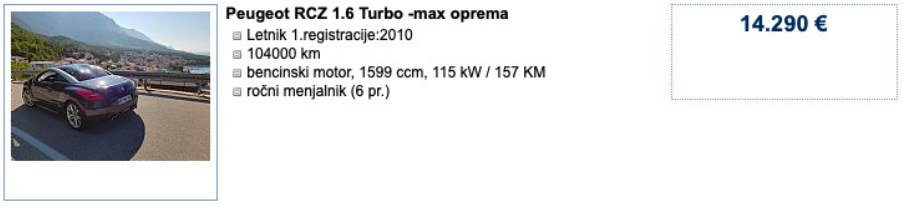


**Fig. 1.** Pimer oglasa s podatki za ekstrakcijo.

## 2 Implementacija metod

Za branje html strani in nato parsanje podatkov smo uporabili knjižnico BeautifulSoup. Pri parsanju z regex-om smo v projekt vključili knjižnico "re". V dokumentaciji za BeautifulSoup je bila podana razlaga o kombiniranju teh dveh knjižnic. Za parsanje z Xpath pa smo vključili še knjižnico lxml.

Za Overstock stran smo z regex-om in xpath-om ekstraktali podatke za vse oglase naenkrat. Namreč, ekstraktali smo vse npr. "Title" atribute za izdelke na strani in jih shranili v tabelo (array). Enako smo storili za vse ostale atribute za izdelke. Na koncu smo šli enkrat z zanko čez tabelo, da smo lepo podatke grupirali in izvozili v json formatu. Isti princip smo uporabili za avto.net pri uporabi regex-a in xpath-a, saj smo procesirali podatke za vse oglase na strani. Za RtvSlo stran je bila zadeva nekoliko preprostejša, saj je bilo enostavneje dostopati do elementov z regex in xpath.

## 3 RoadRunner

Implementacije smo se lotili tako, da smo naprej pregledali literaturo. Zelo nam je pomagal podrobnejši opis roadrunner algoritma in pojasnilo o "tag mismatch" iz magistrskega dela [1]. Znotraj tega dela so bile slike s primeri.

Implementacije smo se lotili naprej tako, da smo iz html-a odstranili javascript, css in odstranili komentarje. Želeli smo imeti samo značke in čist tekst. Naslednja stvar je bila ustvariti drevo. Hoteli smo imeti en koren in več vej oz. otroke za posamezno stran. Nato bi se po vejah sprehodili in iskali razlike med dvema drevesoma.

Na začetku je bil "body" značka označena za koren drevesa. Zato ker imamo dve strani iz iste domene za primerjat, predpostavimo da so nekatere stvari nespremenjene kot npr. pri avtonet sta na obeh straneh enak zgornji menu. Zato smo pogledali prvi nivo otrok tega korena in ohranili tiste enake značke s statičnimi imeni (npr. meni, header..). Nato smo se morali spustiti v vse tiste nivoje otrok, kjer so bile razlike (npr. seznam). Implementacija se nahaja v github projektu.

Ne moremo trditi, da smo zadovoljni z implementacijo našega algoritma. Namreč ponekod so bile težave pri "tag mismatch"-ih, recimo cross reference iskanje ni implementirano in na splošno bi morda bilo bolje izboljšati celoten algoritem.

## 4 Outputs

Vsi generirani podatki v json obliki so v mapi output. Pridobimo jih na standardni izhod tako da izberemo datoteko $regex\_parser, xpath\_parser, roadrunner$ in zaženemo z "Run".

## Literatura

1. Schlyter, E.: Structured data extraction (2007)