Ekstrakcija podatkov - 2. IEPS poročilo

Klobučar Rok, Mihelič Mohor Luka, Ostovršnik Anja

Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

Abstract. Poročilo obravnava tri metode ekstrakcije podatkov s spletnih strani: uporaba regularnih izrazov, XPath in avtomatski spletni algoritem RoadRunner. Podrobno so opisani postopki implementacije za vsako metodo, vključno z izbranimi spletnimi stranmi in identifikacijo podatkovnih elementov. Poudarjene so prednosti in slabosti vsake metode, pri čemer se izpostavlja potreba po posodabljanju izrazov pri uporabi regularnih izrazov in XPath, ter kompleksnost in odvisnost od umetne inteligence pri RoadRunner algoritmu. Kljub izzivom je prednost slednjega pristopa v avtomatiziranem ustvarjanju ovojnic za ekstrakcijo podatkov.

Keywords: ekstrakcija podatkov · brskalnik · svetovni splet

1 Uvod

V poročilu bomo predstavili našo idejo in postopke implementacije ekstrakcijskih metod. Ubrali smo tri metode:

A) Implementacija z uporabo regularnih izrazov,

B) Implementacija z uporabo XPath,

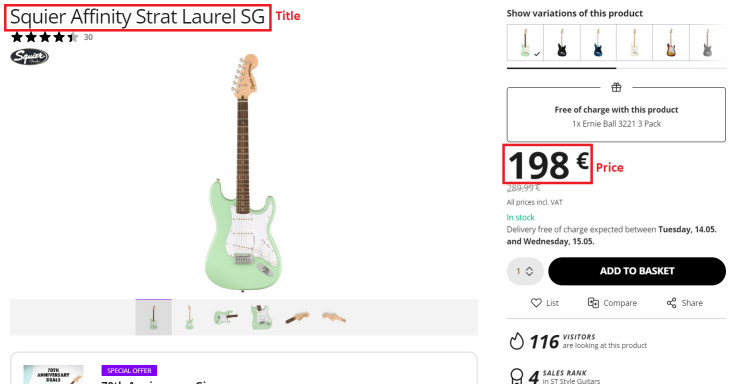
C) Implementacija z uporabo avtomatskega spletnega algoritma za ekstrakcijo RoadRunner

Seznanili smo se z že obstoječimi projekti ter njihovo uporabo in ustrezno literaturo.

2 Opis izbranih spletnih strani

Podatke smo ekstrahirali iz šestih spletnih strani. Od tega so bile vnaprej podane štiri strani: dva članka iz spletne strani rtvslo.si in dva seznama produktov iz spletne strani overstock.com. Za zadnji dve spletni strani smo si izbrali dve strani spletne glasbene trgovine thomann.de. Ekstrahirani podatki so vsebovali naziv, ceno ter opis izdelka, ki se nahaja nižje na spletni strani.

Fig. 1. Primer spletne strani



3 Implementacija z uporabo regularnih izrazov

Pri prvi metodi smo se lotili pridobivanja podatkov z uporabo regularnih izrazov, ki smo jih prilagodili za vsako spletno stran posebej. Funkcijo za ekstrahiranje smo zapisali v datoteki A.py. Prejeto vsebino smo dodatno obdelali, da smo odstranili nepotrebne HTML oznake in morebitne odvečne podatke. Urejene podatke smo nato pretvorili v JSON format, kar izboljša berljivost in preglednost.

Fig. 2. Regularni izrazi spletne strani overstock.com


```
"Title": "10-kt. Seven Diamond Ladies Heart Ring (0.08 TW)",
"ListPrice": "$149.00",
"Price": "$69.99",
"Saving": "$79.01",
"SavingPercent": "53%",
"Content": "This ladies fashion ring dazzles\nwith hearts and diamonds. The gold band is crafted into delicate, open hearts.\nSeven brilliant-cut diamonds add a bit of sparkle. "
```


Fig. 3. Regularni izrazi spletne strani rtvoslo.si


```
"Title": "Audi A6 50 TDI quattro: nemir v premijskem razredu",
"Subtitle": "Test nove generacije",
"Lead": "To je novi audi A6. V razred najdra\u017ejih in najbolj premijskih \u017erebcev je vnesel nemir, \u0161le preden je sploh zapeljal na parkirni prostor, rezerviran za izvr\u0161nega direktorja. ",
"Author": "Miha Perljak",
"PublishTime": "28. december 2018 ob 08:51",
"Content": "class=\"Body\">\nSamo poglejte njegovo masko \u2013 to ogromno satovje z radarji na takem polo\u017eaju, da se ti na avtocesti tudi pri 120 km/h vsi spo\u0161tljivo umikajo, saj so prepri\u010d
```

Fig. 4. Regularni izrazi spletne strani thomann.de


```
"Title": "Squier Affinity Strat Laurel SG \u00e2\u20ac\u201c Thomann International",
"Description": "Electric Guitar  Body: Poplar, Neck: Maple, Neck profile: C, Fingerboard: Laurel, Perloid dot fingerboard inlays, Fretboard radius: 241 mm (9.5\"), Scale:
"Price": "199"
```


4 Implementacija z uporabo XPath

Pri drugi metodi smo izvajali ekstrakcijo podatkov z uporabo XPatha. Za vsak par spletnih strani smo znotraj datoteke B.py implementirali funkcijo, ki je odgovorna za pridobivanje željenih podatkov. Kot vhodni podatek je sprejeta HTML datoteka v obliki niza, iz katere z uporabo knjižnice lxml in XPath izrazov izluščimo željene vsebine.

Fig. 5. XPath izrazi spletne strani overstock.com


```
"Title": "10-kt. Seven Diamond Ladies Heart Ring (0.08 TW)",
"listPrice": "$149.00",
"price": "$69.99",
"saving": "$79.01",
"savingPercent": "53%",
"content": "This ladies fashion ring dazzles\nwith hearts and diamonds. The gold band is crafted into delicate, open hearts.\nSeven brilliant-cut diamonds add a bit of sparkle. "
```


Fig. 6. XPath izrazi spletne strani rtvslo.si


```
"Title": "Audi A6 50 TDI quattro: nemir v premijskem razredu",
"SubTitle": "Test nove generacije",
"Lead": "To je novi audi A6. V razred najdra\u017ejih in najbolj premijskih \u017erebcev je vnesel nemir, \u0161e preden je sploh zapeljal na parkirni prostor, rezerviran za izvr\u0161nega direktorja. ",
"Author": "Miha Merljak",
"PublishTime": "28. december 2018 ob 08:51",
"Content": "\nSamo poglejte njegovo masko \u2013 to ogromno satovje z radarji na takem polo\u017eaju, da se ti na avtocesti tudi pri 120 km/h vsi spo\u0161tljivo umikajo, saj so prepri\u010dani, da gre za
```

Fig. 7. XPath izrazi spletne strani thomann.de


```
"Title": "Squier Affinity Strat Laurel SG \u2013 Thomann International",
"Description": "Electric Guitar  Body: Poplar, Neck: Maple, Neck profile: C, Fingerboard: Laurel, Perloid dot fingerboard inlays, Fretboard radius: 241 mm (9.5\"), Scale:
"Price": "199\u00a0\u20ac"
```


5 Implementacija z uporabo avtomatskega spletnega algoritma za ekstrakcijo RoadRunner

Idejo za našo implementacijo smo dobili iz učbenika Web Data Mining[1]. Algoritem primerja podane strani in išče podobnosti. Iz teh podobnosti sproti generira ovojnico. Vmes lahko pride do razlik v vsebini ali oznaki. RoadRunner se rekurzivno sprehodi čez DOM strukturo preko značk, kjer je priporočeno sprva odstraniti redundantne značke brez semantične vsebine, kot so na primer za prehod v novo vrstico ali vnosi skript. Sproti iz ovojnic obeh izbranih spletnih strani se rekurzivno sprehajamo čez značke in sproti shranjujemo indeks trenutnih ovojnic, kjer je potrebno poskrbeti za značke seznamov.

Psevdokoda algoritma, kjer sta a in b vozlišči drevesa:

```
def RoadRunner(a, b):
    i = 0
    while i < len(a) or i < len(b):
        if eno izmed vozlisc nima vec elementov:
            # Dodaj manjkajoce elemente kot "?"
        elif ujemanje oznak:
            if element je seznam:
                a[i] = iterator_matching(a, b)
            else:
                a[i] = RoadRunner(a[i], b[i])
        elif neujemanje nizov:
            a[i] = "#PCDATA"
        elif ujemanje nizov:
            pass
        elif neujemanje oznak:
            # Navkrizno ujemanje, dodaj ustrezne elemente kot "?"
        return a

def iterator_matching(a, b):
    if len(a) == len(b):
        for i=0:len(a):
            a[i] = RoadRunner(a[i], b[i])
    else:
        if len(a) > len(b) and a[len(a) - 1] != a[len(a) - 2]:
            # Primerjamo istolezne elemente dreves, ostale dodamo kot "?"
            # Podobno storimo, ce do situacije pride v drugem drevesu
        elif n1_len > 0 and n2_len > 0:
```

```
28              # Dodaj manjkajoce elemente kot "+"
29          elif (len(a) == 0 or len(b) == 0) and not len(a) ==
    len(b):
30              # Dodaj manjkajoce elemente kot "  *"
31      return a
```

6 Težave

Med izzive, s katerimi smo se srečali, lahko izpostavimo precej zahtevno in včasih neintuitivno urejanje bs4 drevesa. Težave nam je povzročala napačno prebrana ali celo neprebrana cena izdelka. Prav tako smo imeli težave pri vnašanju iskanja vgnezdenih značk v regularnih izrazih. Težave smo imeli pri RoadRunner implementaciji, ki nam ga ni uspelo dokončati v celoti.

7 Rezultati

Metode imajo med seboj različne prednosti in slabosti. Implementacija z uporabo regularnih izrazov ali XPath izrazov je preprosta in hitra, vendar je njihova pomanjkljivost potreba po stalnem posodabljanju izrazov, če se spletna stran spremeni. Pri RoadRunner algoritmu je težava, da brez pomoči umetne inteligence težko ugotovimo, kateri podatki so dejansko pomembni; potrebne so tudi določene hevristike, ki morda delujejo bolje na nekaterih spletnih straneh kot na drugih. Prav tako je težko določiti pomen avtomatizirano pridobljenih podatkov brez pomoči umetne inteligence. Poleg tega zahteva bistveno boljše razumevanje ekstrahiranja podatkov in znanja programiranja. Kljub temu je prednost tega pristopa očitna, saj ni potrebe po izdelavi ovojnice za vsako stran posebej; ta se ustvari samodejno.

References

1. Bing Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Data-Centric Systems and Applications, 2nd edition, Springer, 2013