Tehničko veleučilište u Zagrebu

Završni rad 999

Računalna obrada prirodnih jezika

Fran Raknić

Zagreb, svibnja 2016.

Sadržaj

[Uvod 3](#__RefHeading___Toc333_1326856321)

[Prikupljanje podataka 4](#__RefHeading___Toc335_1326856321)

[Scrapy 4](#__RefHeading___Toc337_1326856321)

[Arhitektura sustava 5](#__RefHeading___Toc200_166527635)

[Spiders 5](#__RefHeading___Toc202_166527635)

[Facebook Graph API 6](#__RefHeading___Toc252_17155863)

[Implementacija u Pythonu 6](#__RefHeading___Toc254_17155863)

[Natural language toolkit – Python biblioteka 8](#__RefHeading___Toc339_1326856321)

[Opojavničenje / Tokenizing 8](#__RefHeading___Toc341_1326856321)

[NLTK Tokenizer modul 9](#__RefHeading___Toc343_1326856321)

[TweetTokenizer 9](#__RefHeading___Toc345_1326856321)

[Lematizacija / Stemming 11](#__RefHeading___Toc347_1326856321)

[Popis literature 12](#__RefHeading___Toc349_1326856321)

# Uvod

# Prikupljanje podataka

U kontekstu ovog rada prikupljanje podataka odnosi se na dohvaćanje podataka s web mjesta (eng. web scraping), koje je podvrsta rudarenja podataka (eng. data mining). Prilikom prikupljanja podataka na taj način koriste se programi koji na temelju obrazaca pretražuju određeno web mjesto i dohvaćaju podatke koji odgovaraju datom obrascu. Sve programske implementacije *web scrapera* automatizirano pretražuju web mjesta i dohvaćaju sadržaj prema kojem postoji nekakav interes. Obrazac po kojem se prikupljaju podaci najčešće se određuje prema HTML strukturi web mjesta u kombinaciji sa regularnim izrazima (eng. regular expressions). Loša ili promjenjiva struktura HTML-a i sve više dinamičkih web stranica i sadržaja otežavaju a ponekad i onemogućuju dohvat podataka. Program koji se koristi za ovakvu vrstu prikupljanja podataka naziva se web robot (eng. web crawler), kada su dohvaćeni podaci poveznice na druga web mjesta ili stranice koje program prati prema nekim pravilima onda se takav program naziva web pauk (eng. web spider). Slična programska rješenja koriste internetske tražilice koje prikupljaju podatke u svrhu indeksiranja web mjesta.

Za izradu programa za prikupljanje podataka korišten je programski jezik Python 2.7 i njegov modul Scrapy. Python je visoki programski jezik opće namjene a glavni naglasak je na čitljivosti koda i jednostavnosti sintakse. Važna značajka Pythona je interaktivno izvođenje programskog koda što je posebno korisno u izradi programa za web scraping.

## Scrapy

Scrapy je razvojna cjelina za izradu web robota i izvlačenje strukturiranih podataka, napisan je u Python programskom jeziku. Može se koristiti za niz aplikacija kao što je rudarenje podataka, arhiviranje i procesiranje podataka, automatski monitoring i testiranje. Dizajn razvojne cjeline omogućuje da koncentracija razvoja bude na izradi pravila po kojima se dohvaćaju podaci. Sustav je modularan tako da je moguće koristiti dodatne ekstenzije i module na dohvaćenim podacima. Službeni održavatelj i developer je Scrapinghub Ltd. koji licencira Scrapy pod BSD licencom, iako na razvoju sudjeluje 221 developer[Github repozitorij]. Trenutna stabilna verzija Scrapy-a je 1.2.1 objavljena 21.10.2016.

### Arhitektura sustava

Svaka komponenta zadužena je za određeni dio toka podataka unutar sustava.

* Engine

Centralna komponenta Scrapy-a, zadužena za kontrolu toka podataka kroz ostale komponente. Postavlja okidače za određene događaje unutar sustava.

* Scheduler

Zaprima zahtjeve (eng. requests) od engina i određuje redosljed izvođenja te ih vraća enginu.

* Downloader

Komponenta zadužena za dohvat web stranica, nakon dohvata vraća ih enginu da bi bili prosljeđeni paucima.

* Spiders

Pauci su klase izrađene od strane korisnika, zadužene za ekstrakciju podataka ili postavljanje novih requestova.

* Item pipeline

Komponenta zadužena za daljnje procesiranje podataka (itema) koje parsiraju spideri. Uobičajena namjena im je validacija, čišćenje i pohranjivanje podataka (primjerice u bazu podataka).

Osim navedenih komponenti moguće su implementacije eng. Downloader middlewares i Spider middlewares. Downloader middlewares su specifične kuke (eng. hooks) koje se nalaze između Engine i Downloader komponenti i procesiraju zahtjeve (eng. requests) od Engine-a prema Downloader-u i odgovore (eng. responses) koje idu od Downloader-a prema Engine-u. Ove funkcionalnosti su korisne kada je potrebno izmjeniti response prije nego što je poslan Spider komponenti ili zabranjivanje slanja određenih requestova.

### Spiders

Pauci (eng. Spiders) su klase koje definiraju kako će podaci sa određenih web stranica biti dohvaćeni, uključujući i kako će poveznice unutar tih stranica biti praćene. Ciklus dohvaćanja počinje sa inicijalnim zahtjevima na (eng. requests) zadane URL-ove, te određivanje callback funkcije koja će biti pozvana kada spider primi odgovor (eng. response). Unutar callback funkcije parsira se vraćeni odgovor te vraća parsirane podatke u obliku Item objekta, Request objekta ili iterabilnog tipa koji sadrži neke od navedenih objekata. U callback funkciji određujemo elemente web-a koji želimo dohvatiti. Za pronalazak željenih elemenata moguće je koristiti CSS selektore, Xpath (eng. XML Path Language) ili vanjske biblioteke za parsiranje.

## Facebook Graph API

Aplikacijsko programsko sučelje (eng. Application Programming Interface, API) je skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima nekog složenog programa. Graph API je osnovni način za dohvat podataka iz Facebook platforme. API je baziran na HTTP-u (hypertext transfer protocol) koji se koristi za izradu upita prema podacima kao što su objave, reklame, fotografije i niz drugih zadaća ovisno o potrebi aplikacije. Graph API predstavlja podatke na sljedeći način:

* nodes – objekti kao što su korisnik, fotografija, komentar, stranica
* edges – veze između objekata, fotografije stranice, komentari fotografije
* fields – informacije o objektima kao rođendan korisnika, ime stranice ili fotografije

Budući da je API baziran na HTTP-u moguće je koristiti bilo koji jezik koji sadrži HTTP biblioteku. Za pristup API-ju potreban je dokaz za pristup (eng. access token). Access token je niz znakova koji identificira korisnika, aplikaciju ili stranicu koja pristupa API-ju te ograničava prava pristupa podacima.

### Implementacija u Pythonu

Za dohvaćanje podataka sa API-ja u ovome radu koristiti će se programski jezik Python i HTTP biblioteka urllib2. Osnova korištenja API-ja baziranim na HTTP-u je slanje GET requestova na određeni URL.

[isječak koda funkcije request\_until\_succeed]

Funkcija prima parametar URL (eng. Uniform Resource Locator) te šalje GET zahtjev na taj URL. Ukoliko je odgovor servera 200 (što znači da je zahtjev uspješan) funkcija vraća podatke poslane od strane servera. Ako dođe do greške obavještava korisnika koji URL zahtjev nije uspio i vrijeme greške. Ova osnovna funkcija koristiti će se za sve zahtjeve prema API-ju.

Podaci koje gore prikazana funkcija dohvaća ovise o datom parametru URL. Za dohvaćanje objava i meta-podataka sa neke stranice implementirana je funkcija koja određuje potrebne dijelove URL-a i prosljeđuje niz znakova koji predstavljaju URL funkciji [request\_until\_succeed]. API vraća podatke u JSON (eng. JavaScript Object Notation) obliku. JSON je format za razmjenu podataka, lako je čita i piše te je neovisan o izboru programskog jezika. Koristi se konvencije koje su poznate programskim jezicima iz C obitelji što ga čini idealnim izborom za izmjenu podataka. Za manipulaciju JSON tipom podatka koristi se Python-ov modul *json* koji je dio standardne biblioteke.

[Isječak koda funkcije getFacebookPageFeedData]

Funkcija [getFacebookPageFeedData] vraća dohvaćene podatke u JSON obliku za daljnju obradu. U varijabli [parameters] se sprema broj koji određuje koliko će se objava dohvatiti i [access\_token] – niz znakova koji služi za autorizaciju i autentifikaciju programa koji šalje zahtjeve prema API-ju. Kada se dohvaća veći broj objava API podatke razdvaja po stranicama. Za "listanje" stranica potrebno je slati zaseban zahtjev za iduću ili prethodnu stranu. URL za navigaciju po stranicama predstavljen je u JSON-u u objektu [paging] koji sadrži [next i previous] ključeve sa odgovarajućim vrijednostima u obliku niza znakova.

# Natural language toolkit – Python biblioteka

Natural Language Toolkit skraćeno NLTK je platforma namjenjena obradi prirodnih jezika za programski jezik Python. Sadrži biblioteke za procesiranje teksta, kao što su: klasifikacija, tokenization, stemming, tagging, parsing itd. Također sadrži preko pedeset korpusa i trening modela za strojno učenje. Biblioteka je dostupna na Windows, Mac OS X te Linux/UNIX operacijskim sustavima te je licencirana pod Apache 2.0 licencom što znači da je kod otvoren te korištenje besplatno. Posljednja stabilna verzija je 3.2.1 distribuirana 9. travanja 2016. NLTK zahtjeva Python verzije 2.7 ili 3.2, neke funkcionalnosti za strojno učenje dodane u 3.0 verziji zahtjevaju Python 3.2 te neće raditi na 2.7 verziji.

Natural Language Toolkit, skraćeno NLTK je niz biblioteka i programa za simboličko i statističko obrađivanje pomoću programskog jezika Python. Natural Language Toolkt uključuje ogledne primjerke te je popraćen opsežnom dokumentacijom i uključuje knjigu objašnjenja fundamentalnih načela iza obrađivačkih zadataka podržanih od strane ovih alata. Pretežito je namjenjen učenju o računalnoj obradi prirodnog jezika ili za istraživanja prirodnog jezika.

## Opojavničenje / Tokenizing

Opojavničenje je proces određivanja lingvističkih jedinica (rečenice, riječi, znakovi...). Opojavničenje ili tokenizacija stavlja korpus u stanje u kojem su sve riječi-pojavnice identificirane i eksplicitno obilježene. Najjednostavniji i prilično učinkovit pristup je razdvajanje pojavnica znakom za obilježavanje razmaka tj. bijelinama. Automatizirati ovakav proces je prilično jednostavno, većina programskih jezika sadrži standardne biblioteke za string strukturu podataka koje sadrže funkcije za osnovnu obradu teksta (split, replace, upper, lower). Međutim, opojavničenje/tokenizacija u složenijim pristupima, primjerice pojavnice se mogu sastojati od dvije ili više riječi. Na primjer, datum *15. svibnja* ili *15. 5.* bi mogao biti obrađivan kao jedna pojavnica pa ranije spomenuti jednostavni pristup nebi mogao tokenizirati. Tekst koji se obrađuje može se opojavničiti više puta da bi se dobili željeni rezultati. Nakon dobivanja tokena koji su riječi ponekad je potrebno spojiti više riječi koje, gotovo uvijek, imaju isto značenje, primjerice „hard drive“. Nakon tokeniziranja manjih lingvističkih jedinica kao što su riječi ili datumi potrebno je označiti i rečenice. Postupci segmentacije na rečenice su složeni iz razloga što su oznake rečenične interpunkcije često višeznačne. Na primjer, točka može stajati uz redni broj, kraticu ili na kraju rečenice. Podaci također mogu dolaziti označeni nekim od jezika za označavanje (eng. *markup languages*) primjerice u HTML-u ili XML-u. Prije opojavničenja teksta prirodnog jezika potrebno je odvojiti jezik označavanja od teksta koji želimo obrađivati. Budući da su jezici označavanja (formalni, umjetni, kontrolirani?)uvelike standardizirani taj proces je olakšan radi mogućnosti korištenja gotovih riješenja koja uklanjaju elemente kao što su tagovi. U slučajevima kada su podaci označeni za aplikacije i procesore treće strane (eng. *third-party*) potrebno je proučiti logiku označavanja te izraditi vlastito riješenje koje uklanja ili odvaja tekst od jezika označavanja. Posebno teško je opojavničenje jezika u scriptio continua pismu gdje se riječi ne razdvajaju razmacima ili znakovima (Kineski, Thai, staro-Grčki).

Izazovi opojavničenja teksta komunikacije na internetu, posebice društvenim mrežama, uvelike ovise o ciljevima obrade i platformi na kojoj se komunikacija odvija. Svaka društvena mreža ili internetski servisi kao chat sobe imaju neke posebne znakove ili standarde. Primjerice „Tweeter“ koristi znak # (eng. *hashtag*) nakon kojeg, bez razmaka, slijedi riječ koja označava širu temu ili područje objave. Ovisno o ciljevima, aplikacija za obradu može tretirati takve tokene kao riječi, posebnu kategoriju ili odbacivati.

### NLTK Tokenizer modul

NLTK platforma sadrži Tokenizer paket. Bazna klasa svih tokenizera/opojavnjivača je „tokenize“, instanciranjem svojih objekata ili korištenjem postojećih modula sve klase proširuju „tokenize“. Također nasljeđuju tokenize metodu koja će biti opisana za svaki pod-modul posebno.

### TweetTokenizer

Prilagođen opojavničenju Tweet objava, fleksibilan i lako podesiv. Tuple regex\_strings definira listu regularnih izraza. Regularni izrazi su, po redu, kompajlirani u jedan regexp objekt naziva word\_re. Metoda tokenize prima dva dodatna argumenta, preserve\_case sa zadanom vrijednosti True, te reduce\_len sa zadanom vrijednosti False. U slučaju da korisnik prvi parametar postavi na False, metoda će sve tokene postaviti na mala slova dok će drugi parametar, ukoliko je True smanjiti sva uzastopna ponavljanja znaka na tri ponavljanja. Na primjer *jaaaaako* pretvara u *jaaako.*

EMOTICONS = r"""

(?:

[<>]?

[:;=8] # eyes

[\-o\\*\']? # optional nose

[\)\]\(\[dDpP/\:\}\{@\|\\] # mouth

|

[\)\]\(\[dDpP/\:\}\{@\|\\] # mouth

[\-o\\*\']? # optional nose

[:;=8] # eyes

[<>]?

|

<3 # heart

)"""

From nltk.tokenize import TweetTokenizer  
  
Tobjekt = TweetTokenizer()

niz = „Ovo je niz emotikona i :D :-P D <3“

Tobjekt.tokenize(niz)

Kod tipa teksta kao što su neformalne objave na društvenim mrežama, taj proces se komplicira. Primjerice neki korisnici koriste hrvatske dijakritike, neki ih pišu u obliku ch za ć ili dj za đ. Komunikacija preko društvenih mreža sadržava poveznice na druge stranice koje mogu dolaziti kroz API te kao niz znakova. Pokazati neke od implementacija (NLTK).

NLTK biblioteka nudi više modula za opojavničenje. U načelu svaki modul razdvaja niz znakova u određene lingvističke jedinice. Osnovni zadatak je razdvojiti tekst na riječi, tj. razdvojiti interpunkciju i posebne znakove od jedinica koje imaju značenje. Modul TweetTokenizer sadrži regularne izraze koji prepoznaju tipove kao što su URL, emotikoni te HTML.

## Lematizacija / Stemming

Lematizacija (stemming) je dugotrajnij i teži postupak od opojavničenja. Lematizacija je proces reduciranja flektivnih riječi da bi se dobio korijenski oblik, korijen ne mora nužno biti isti kao morfološki. Za dobivanje oblika riječi hrvatskog jezika moguće je koristiti Hrvatski Morfološki Leksikon[3], za precizniju lematizaciju podataka dobivenih s društvenih mreža ili bilo kojeg izvora neformalne komunikacije među korisnicima bilo bi potrebno napisati vlastita pravila ili izraditi leksikon za specifično područje. Primjeri implementacije...

## Popis literature