# Uvod

# Prikupljanje podataka

Stalno stvaranje novog sadržaja na internetu čini ga neiscrpnim izvorom podataka. U kontekstu ovog rada prikupljanje podataka odnosi se na dohvaćanje prirodnog jezika sa web mjesta. Društvene mreže poput Facebook-a imaju dokumentiran i otvoren sustav za dohvaćanje i objavljivanje sadržaja što olakšava pristup podacima zanimljivim za obradu. Ako web mjesto nema otvoren programatski pristup svojim podacima onda se koristi web struganje (eng. web scraping). Za izradu programa za prikupljanje podataka korišten je programski jezik Python 2.7. Python je visoki programski jezik opće namjene a glavni naglasak je na čitljivosti koda i jednostavnosti sintakse. Važna značajka Python-a je interaktivno izvođenje programskog koda što je posebno korisno u izradi programa za prikupljanje podataka. Za izradu programa za web struganje koristit će se razvojna cjelina Scrapy.

## 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.



Slika Funkcija request\_until\_suceed

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 program se zaustavlja na 5 sekundi prije ponovnog zahtjeva te obavještava korisnika koji URL zahtjev nije uspio i vrijeme greške. Ova osnovna funkcija koristiti će se za sve zahtjeve prema API-ju.

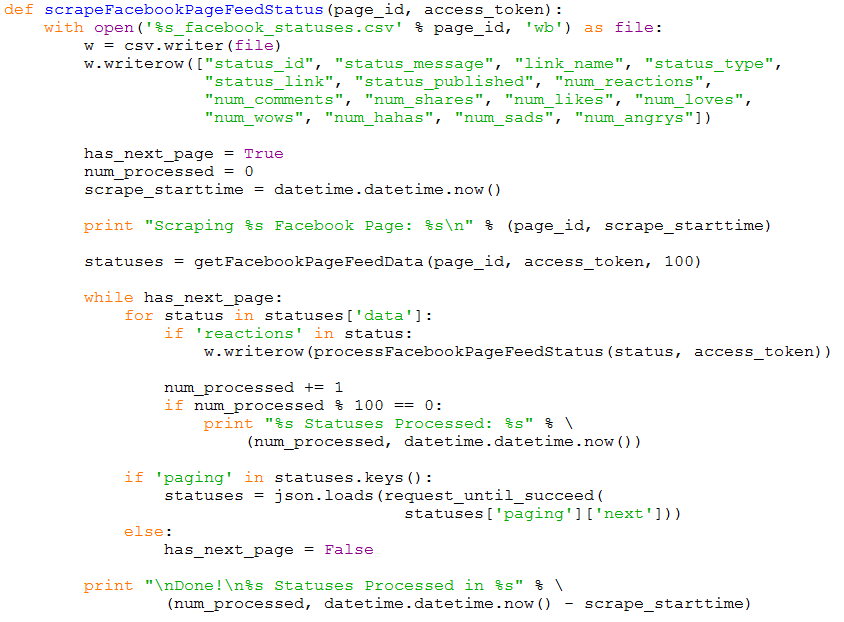
Podaci koje gore prikazana funkcija dohvaća ovise o datom parametru URL. Za dohvaćanje objava i meta-podataka sa neke stranice implementirana je funkcija koja određuje potrebne dijelove URL-a i prosljeđuje niz znakova koji predstavljaju URL funkciji [request\_until\_succeed]. API vraća podatke u JSON (eng. JavaScript Object Notation) obliku. JSON je format za razmjenu podataka, lako se čita i piše te je neovisan o izboru programskog jezika. Koristi 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.



Slika Funkcija 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.

Prikupljeni podaci spremaju se u tekstualnu datoteku za daljnju obradu. Odabrani formati je CSV (eng. Comma Seperated Values), podaci se spremaju u polja koja su razdvojena zerezima ili drugim arbitrarnim znakom. Razlog odabira ovog formata je podrška u raznim programima i programskim jezicima te čitljivost i podrška za posebne znakove. Format je također služi za uvoz i izvoz podataka iz baza podataka što može biti korisno kada se veličina datoteka poveća ili se podaci povezuju sa raznim servisima i aplikacijama. Za rukovanje CSV datotekama u implementaciji za dohvat podataka koristi se modul iz standardne Python-ove biblioteke *csv,* prije unosa dohvaćenih podataka program upisuje zaglavlja koja označuju o kojem meta-podatku se radi.



Slika scrapeFacebookPageFeedStatus

Funkcija [scrapeFacebookPageFeedStatus] koristi navedene funkcije za interakciju sa API-jem i procesuiranjem dohvaćenih podataka, obavještava korisnika o vremenu i broju dohvaćenih podataka te ih sprema u CSV datoteku za daljnju obradu.

Parametar *page\_id* je jedinstveni identifikator stranice sa koje želimo dohvatiti objave. Identifikator se lako može saznati tako da iz internetskog pretraživača posjetimo stranicu te iz URL-a isčitamo identifikator.



Slika Jedinstveni identifikator stranice ili grupe

Objave koje se dohvaćaju obično budu vezane za određenu temu a mogu biti objavljivane od strane korisnika učlanjenih na stranicu ili kao u slučaju portala koji imaju službene Facebook stranice objave dolaze jedino od vlasnika stranice dok korisnici ostavljaju komentare na objave. Prije prikupljanja podataka korisno je "ručno" analizirati potencijalnu stranicu da bi se utvrdile neke osnovne karakteristike kao što je učestalost objava, tematika objava i tip objava (primjerice stranice gdje korisnici postavljaju pitanja na određenu temu).

## Web scraping

Ako podaci koje želimo dohvatiti nisu dostupni na druge načine poput API-ja onda se koristi web scraping. Prilikom prikupljanja podataka na taj način koriste se programi koji na temelju obrasca pretražuju određenu web stranicu i dohvaćaju podatke koji odgovaraju obrascu. Programske implementacije web scrapera automatizirano pretražuju web stranice i dohvaćaju sadržaj koji odgovara kodiranom obrascu. Obrazac po kojem se prikupljaju podaci najčešće se određuje prema HTML strukturi 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 prikupljanje 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). Web robote i web pauke koriste internetske tražilice da bi indeksirale web i omogućile pretraživanje kakvo svakodnevno koristimo.

Izrada programa za web scraping može biti jednostavan zadatak ukoliko se radi o dohvaćanju manje količine podataka sa malog broja web stranica. Kada se radi o većem broju stranica sa kompliciranijim obrascima za dohvaćanje podataka te potrebom da se web stranice provjeravaju za novi sadržaj ili pojavu novih web stranica onda se potrebno pobrinuti da programski kod bude održiv i čitljiv. 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, na razvoju sudjeluje 221 developer[Github repozitorij]. Uz sve prednosti koje pruža korištenje razvojne cjeline, Scrapy također ima mogućnost izvođenja u oblaku. Kada se radi web scraping na velikim web stranicama koje sadrže puno podataka, robot šalje veliki broj zahtjeva prema poslužiteljima što može postati nestabilno kada se koristi osobna internetska usluga. Resursi osobnog računala općenito postaju nedovoljni kada proces prikupljanja podataka zahtjeva veće količine memorije i duže vrijeme izvođenja. Scrapy Cloud omogućuje implementaciju zahtjevnijih web robota bez potrebe održavanja infrastrukture koju takvi web roboti zahtjevaju. Prikupljeni podaci se spremaju u high-availability baze podataka i dostupni su u JSON, XML i CSV oblicima.

### Implementacija Scrapy-Python

# Korpusna lingvistika

Korpusna lingvistika je grana lingvistike koja jezik proučava kroz korpuse. Ova definicija uključuje razvijanje i primjenu tehnika za sastavljanje korpusa. Najjednostavnija definicija korpusa (lat. Tijelo) je skup tekstova pohranjenih u jednu cjelinu. Dakle, korpusna lingvistika je grana koja se služi određenim procedurama i metodama da bi proučavala jezik. Iako su principi korpusne lingvistike postojali i prije pojave računala gotovo je neizbježno koristiti računalo za čitanje, pretraživanje i obradu podataka. U kontekstu ovog rada korpusnu lingvistika bi mogla biti definirana kao proučavanje jezika kroz skup tekstova u obliku čitljivom za računalo. Skup tekstova tj. korpus najčešće je veličine koja nebi dozvoljavala ručno istraživanje nego zahtjeva određene alate da bi se provela kvantativna i kvalitativna analiza podataka[Gatto 2014]. Važno je primjetiti da korpusna lingvistika proučava jezik na temelju primjera stvarne upotrebe jezika za razliku od tradicionalnih istraživanja u kojima se do zaključaka dolazi uglavnom putem introspekcije izvornog govornika.

Kvaliteta i važnost istraživanja korpusa ovisi o izboru korpusa koji istražujemo da bi odgovorili na određena pitanja ili testirali hipoteze. Primjerice, ne mogu se donositi zaključci o cjelokupnom jeziku ako korpus koji istražujemo sadrži samo tekstove iz dnevnih novina. Suptilniji primjer bi bio da prije istraživanja pretpostavimo da je korpus dnevnih novina homogen, dok bi u stvarnosti bilo leksičkih i drugih razlika između sportskog i kulturnog dijela novina. Dakle izbor korpusa ili izrada novog specijaliziranog korpusa za svrhu istraživanja uvelike određuje kvalitetu rezultata istraživanja korpusa.

## Veličina korpusa

Veličina korpusa ovisi o namjeni i izvoru tako da ne postoji standard koji bi određivao veličinu korpusa. Korpusi koji predstavljaju određeni jezik u cjelosti svakako moraju biti veliki dok korpusi koji predstavljaju određeni podskup jezika, tj. specijalizirani korpusi, su relativno male veličine. Pojmovi „veliko“ i „malo“ u kontekstu korpusa su se mijenjali kroz vrijeme. Prvi računalni korpus, Brown Corpus sadržavao je milijun riječi engleskog jezika. Brown Corpus bi mogao spadati u male korpuse po standardima druge generacije općih korpusa kao što je British National Corpus (BNC) koji sadrži oko 100 milijuna riječi engleskog jezika. Korpusi koji se koriste za izradu riječnika, primjerice *Collins Cobuild Bank of English* sadrže 650 milijuna riječi sakupljenih iz pisanog i izgovorenog engleskog jezika.

Preporuka je da korpus bude što veći s mogućnošću rasta a preporuka se temelji na opće poznatoj činjenici da se oko pola vokabulara svih korpusa sastoji od riječi koje se pojavljuju samo jednom (Zipf 1935.). Dakle može se zaključiti da je opća preporuka da korpus bude što veći ako su tekstovi od kojih se korpus izrađuje relevantni za istraživanje.

## Opojavničenje

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.

# Izrada korpusa

## Natural Language Processing Toolkit

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.

# Zaključak

# Literatura

Tony McEnery, Andrew Hardie: „Corpus Linguistics“, Cambridge Texbooks in Linguistics Ujedinjeno Kraljevstvo, Cambridge 2012

Maristella Gatto: „Web as Corpus“, 2014