Contents

[1 Uvod 2](#_Toc474895263)

[2 Prikupljanje podataka 3](#_Toc474895264)

[2.1 Facebook Graph API 3](#_Toc474895265)

[2.1.1 Implementacija u Pythonu 3](#_Toc474895266)

[2.2 Web scraping 6](#_Toc474895267)

[2.2.1 Scraping foruma 7](#_Toc474895268)

[2.3 Baza prikupljenih podataka 11](#_Toc474895269)

[3 Korpusna lingvistika 13](#_Toc474895270)

[3.1 Vrste korpusa 13](#_Toc474895271)

[3.2 Web kao korpus 14](#_Toc474895272)

[3.3 Veličina korpusa 15](#_Toc474895273)

[3.4 Važnost i uloga korpusa 15](#_Toc474895274)

[4 Izrada korpusa 16](#_Toc474895275)

[4.1 Natural Language Processing Toolkit 16](#_Toc474895276)

[4.2 Opojavničenje 17](#_Toc474895277)

[4.2.1 Opojavničenje prikupljenih tekstova 18](#_Toc474895278)

[4.3 Korjenovanje 19](#_Toc474895279)

[5 Klasifikacija teksta 21](#_Toc474895280)

[5.1 Naivni Bayesov klasifikator 21](#_Toc474895281)

[6 Zaključak 22](#_Toc474895282)

[7 Dodaci 22](#_Toc474895283)

[7.1 Razlike između Python 2.x i Python 3.x 22](#_Toc474895284)

[7.2 Bug forum HTML struktura 23](#_Toc474895285)

[8 Literatura 25](#_Toc474895286)

# Uvod

Cilj ovog rada je opisati cijelokupni proces stvaranja i korištenja korupsa. Odabrani tekstovi predstavljaju neformalnu komunikaciju među korisnicima društvene mreže facebook i internetskog foruma bug.hr. Za prikupljanje podataka opisane su i implementirane dvije metode: korištenje API-ja (eng. *Application Programming Interface*) i struganje web stranica. Prikupljeni podaci se spremaju u bazu podataka koja se koristi za sve ostale procese izrade i uporabe korpusa. Količina novih tekstova na webu i utjecaj istih na jezik potakla je mnoge autore (lingviste) da ponovno definiraju što je korpus. Budući da je sav tekst na webu u digitalnom obliku te se tekstovi sa weba koriste za izradu korpusa, moguće je na web gledati kao na korpus.

Nad prikupljenim podacima se primjenjuju fundamentalne metode korpusne lingvistike za što se koristi Natural Language Toolkit. Opisane su dvije metode opojavničenja te su prikazani rezultati opojavničenja prikupljenih tekstova. Demonstrirane su neke funkcionalnosti kao što su prebrojavanja i statističke informacije nad opojavničenim tekstom. Također je prikazana metoda morfološke normalizacije teksta – korjenovanje.

Zadnje poglavlje ovog rada odnosi se na klasifikaciju teksta, koja nemora nužno biti vezana za korpusnu lingvistiku. Algoritam za klasifikaciju koji se koristi je naivni Bayesov klasifikator. Ovaj klasifikator koristi tehniku nadziranog strojnog učenja da bi automatski klasificirao tekst. Za klasifikaciju su odabrani neki od prikupljenih podataka, a korpusne metode kao što je opojavničenje korištene su u implementaciji Bayesovog klasifikatora. Cilj ovog poglavlja je prikazati kako korpusi i korpusne metode mogu biti korišteni izvan lingvističkih istraživanja.

# 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.



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 bazu podataka 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

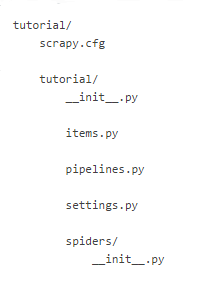
Ako podaci koje želimo dohvatiti nisu dostupni na druge načine poput API-ja onda se koristi web scraping. Prilikom prikupljanja podataka na taj način koriste se programi koji na temelju obrasca pretražuju određenu web stranicu i dohvaćaju podatke koji odgovaraju obrascu. Programske implementacije web 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 za preuzimanje u JSON, XML i CSV oblicima.

### Scraping foruma

Određivanje obrasca prikupljanja podataka postiže se navigacijom po strukturi (X)HTML dokumenta. Scrapy razvojna okolina nudi dva ugrađena načina za navigaciju strukturom dokumenta: CSS i XPath. Budući da se izgled HTML stranica definira CSS-om (eng. *Cascading Style Sheets*) potreban je način da se odredi koje elemente HTML-a će označavati kako bi ih se oblikovalao na željeni način. Ovaj isti princip se može koristiti za odabir dijela HTML-a koji želimo dohvatiti. Drugi način, XPath, je jezik za upite koji odabire članove iz XML dokumenata a bazira se na predstavljanju XML dokumenta kao stabla.

Stvaranje novog projekta razvojnog okruženja Scrapy pojednostavljeno je *command-line* alatima. Korištenjem alata program izrađuje direktorij koji sadrži kostur projekta sa početnim datotekama potrebnim za rad scrapera.



Slika 5 Početni direktorij Scrapy projekta

Osnovna komponenta svakog Scrapy projekta je pauk koji posjećuje i dohvaća podatke sa željenih web stranica. Pauk se definira klasom koja nasljeđuje iz klase *Spider* te implementira ime pauka, dozvoljene domene koje pauk posjećuje te početne URL-ove od kojih pauk počinje pretraživati. Osnovna metoda *Spider* objekta parsira sadržaj koji je dohvaćen s URL-a koji su zadani kao početni.



Slika Klasa ForumSpider

Varijabla *teme* u metodi *parse* nalazi poveznice na sve teme dostupne na forumu. Budući da je cilj pauka da prikupi sve postove, metoda *parse* posjećuje sve poveznice te za svaku poveznicu poziva callback funkciju *parse\_tema.* Ključna riječ *yield* u Python programskom jeziku vraća iterabilne objekte nad kojima se iteriacija radi samo jednom. U ovom slučaju to su objekti tipa *Request* sa argumentima URL-a i funkcije koja će se pozvati kada se taj URL posjeti. *Parse* metoda se koristi kao prvi nivo scrapanja domene. Drugi nivo se odnosi na same pod teme koje pišu korisnici foruma. Metoda *parse\_tema* posjećuje sve pod teme te za svaku od njih poziva callback metodu *parse\_post.* Uz uobičajeni način dohvaćanja URL-ova koji vode do samih tema, kod scrapinga se često treba pobrinuti oko straničenja (eng. pagination). Na forumu koji je odabran za dohvaćanje podataka paginacija je implementirana kao na slici:



Slika Paginacija Bug foruma

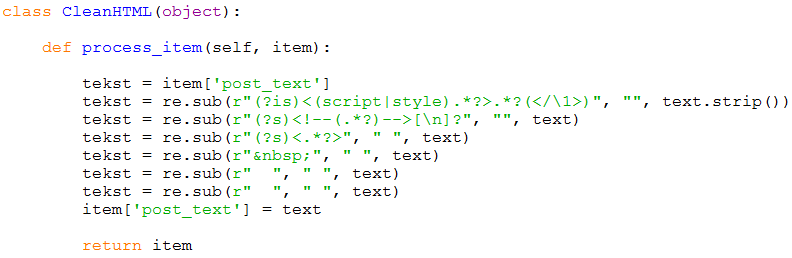
Svaki broj stranice sadrži poveznicu na forum koju obilježava kao i poveznica na sljedeću stranicu. Ukoliko ta poveznica postoji tj. ako URL nije prazan, isti se posjećuje te ponovno poziva metodu *parse\_tema* koja se izvodi dok se ne posjete sve pod-teme. Zadnji korak u dohvaćanju podataka je izrada klase koja će predstavljati „predmet“ koji dohvaćamo. Korištenjem alata za izradu kostura projekta, među ostalim datotekama, napravljena je i datoteka *items.py*. U toj datoteci se definira klasa koja će predstavljati objekt koji dohvaćamo. Klase u *items.py* datoteci zamišljene su kao spremnici za podatke koji će se popuniti za vrijeme rada pauka. Metoda *parse\_post* koristi klasu koju smo definirali u datoteci *items.py* te definira kojim sadržajem će se polja popuniti. Budući da se prikupljaju objave na forumu, klasa koja predstavlja „predmet“ sadrži sljedeća polja: tekst objave, datum objave, temu objave i URL poveznicu na objavu.

Nakon što je pauk prikupio „predmet“ često ga je potrebno dodatno obraditi da bi se doveo u željeno stanje ili oblik. Klase u datoteci *pipelines.py* predstavljaju komponente cjevovoda (eng. *pipeline*) koje implementiraju metodu *process\_item*. Metoda se poziva za svaki prikupljeni „predmet“ te provodi određeno obrađivanje ili odlučuje trebali predmet biti odbačen. Tipične primjene cjevovoda su:

* Čišćenje teksta od jezika označavanja (HTML-a)
* Validacija dohvaćenih podataka
* Provjera jesu li podaci već prikupljeni
* Spremanje predmeta u bazu podataka

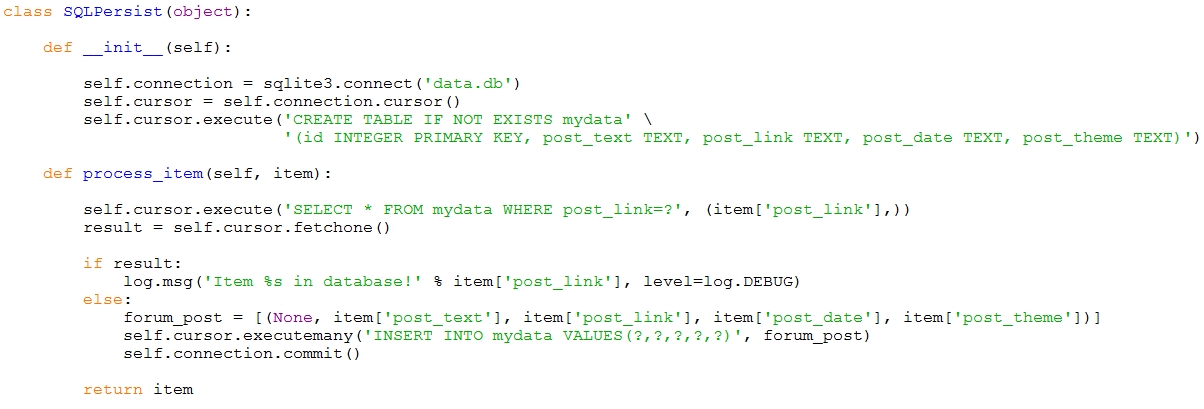
Prva klasa cjevovoda obrađuje oblik u kojem je prikazano i prikupljeno vrijeme objave. Prije obrade podataka vrijeme objave u „predmetu“ je znakovni niz u obliku koji prikazuje dan u tjednu, datum i vrijeme. Iako je ovaj oblik zapisivanja vremena čitljiv ljudima, korisnicima foruma, za potrebe daljnje obrade podataka potreban je oblik koji je strojno čitljiv. Klasa *DateParse* definira metodu kojom se vrijeme objave pretvara u oblik pogodan za daljnju obradu. Konačni oblik vremena objave je *2014-04-11 10:34*.

Predmet koji je definiran u *items.py* datoteci sadži polje koje predstavlja tekst objave na forumu. Osim teksta koji je napisao korisnik, sustav foruma dodaje HTML tagove koji prikazuju elemente teksta na određen način, primjerice prikaz teksta u numeriranoj listi. Budući da je namjena prikupljanja teksta analiza prirodnog jezika, HTML tagovi koji služe za određeni prikaz teksta nisu poželjni. Za pročišćavanje teksta od HTML tagova koriste se regularni izrazi koji su definirani u klasi *CleanHTML.*



Za korištenje regularnih izraza potrebno je uključiti modul *re* koji je sadržan u standardnoj Python-ovoj biblioteci. Funkcija *re.sub* prima regularni izraz, tekst nad kojim ga primjenjuje te znakovni niz kojim će tekst biti zamjenjen ukoliko se poklapa sa regularnim izrazom. Čišćenje se izvodi u koracima, prvi korak je brisanje eng. *inline* skripti i stilova, drugi brisanje HTML komentara, treći brisanje tagova te brisanje HTML znakova i razmaka.

Zadnja komponenta u cjevovodu izrađuje tablicu za pohranjivanje „predmeta“ u bazu podataka te pohranjuje podatke.



Kada se objekt *SQLPersist* instancira, program se se spaja na bazu, ako tablica za podatke ne postoji program izvršava upit koji izrađuje tablicu. Prije spremanja prikupljenog „predmeta“ provjerava se postoji li predmet u bazi podataka, ako postoji program javlja o kojem URL-u se radi te odbacuje „predmet“. Jednostavan i učinkovit način za provjeru duplikata je URL poveznica jer ako je već u bazi znači da je sadržaj na koji taj URL pokazuje već posjećen i dohvaćen. Sve klase, tj. komponente cjevovoda izvode se sekvencijalno, redom koji definiramo u konfiguracijskoj datoteci *settings.py.* Moguće je koristiti sličan princip da bi se obradili URL-ovi prije nego što ih pauk posjeti.

## Baza prikupljenih podataka

Baza podataka koja se koristi za pohranu je SQLite. SQLite je DBMS koji je sadržan u C biblioteci te je ugrađeni (eng. embedded) sustav što znači da nema odvojeni proces. Razlog odabira ove baze podataka je što nema potrebe za administracijom, definicija i podaci se spremaju u jednu datoteku te podrška u gotovo svim programskim jezicima. Nedostatak korištenja SQLite baze podataka je nemogućnost paralelnog pisanja u bazu podataka iz više programa[Službena dok. SQLite].

Programski jezik Python sadrži modul *sqlite3* koji služi za interakciju sa bazom te je sadržan u standardnoj biblioteci. Ovaj modul koristi se u oba načina prikazana u poglavlju Prikupljanje podataka. Takav način interakcije programa s bazom podataka može se koristiti u manjim aplikacijama sa jednostavnim shemama baze podataka. Za kompleksnije sheme i učestalije korištenje baze obično se implementira eng. *Database abstraction layer* koji olakšava razvoj, odvaja logiku programa koji predstavlja podatke i čini programski kod čitljivijim i lakšim za održavanje. Takav sloj realizira se korištenjem ORM-a (eng. *Object Relational Mapper*) koji podatke u bazi predstavlja kao instance objekata. Jedan od primjera za programski jezik Python je SQLAlchemy[Službena dok. SQLAlchemy]. U kontekstu ovog rada nije bilo potrebe koristiti ORM radi relativno jednostavnog korištenja baze podataka ali ukoliko bi se broj izvora podataka povećao, shema baze bi postala kompleksnija čime bi se stvorila potreba za ORM-om.

Baza prikupljenih podataka sastoji se od jedne tablice koja sadrži šest polja. Odabran je ovako jednostavan pristup jer prikupljene objave dolaze iz dva izvora te osim teksta same objave postoje i drugi podaci koji nisu isti za oba izvora. Primjerice objave sa facebook društvene mreže sadrže broj „lajkova“, a objave sa foruma „zahvale“. Ovakvi meta-podaci mogu biti korisni ako se analiziraju podaci iz jednog izvora, za podatke iz više izvora od kojih svaki ima specifične meta-podatke bilo bi potrebno analizirati u kakvoj su vezi meta-podaci iz različitih izvora. Svaka prikupljena objava sadržava tekst, link, temu, vrijeme objave i vrijeme prikupljanja. Izvođenje programa iz poglavlja 2.1.1 Implementacija u Pythonu, ovisno o broju objava u grupi traje između 5 i 10 minuta. Za izvođenje programa iz poglavlja 2.2.1 Scraping foruma potrebno je XX sati u kojima je poslano XX HTTP zahtjeva i posjećeno XX web stranica na domeni bug.hr

# 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 koje provodi nad korpusima 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 korpusna lingvistika bi mogla biti definirana kao proučavanje jezika kroz skup tekstova u obliku čitljivom za računalo. Skup tekstova tj. korpus najčešće je veličine koja ne bi 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.

## Vrste korpusa

Općenito korpusi su dizajnirani za određenu namjenu u istraživanju jezika. Vrste korpusa se mogu podjeliti s obzirom na namjenu. Prva razlika između korupsa su općejezični i specijalizirani korpusi, također važna podjela prema vremenu su sinkronijski i dijakronijski te jednojezični i višejezični. Općejezični korpusi sadrže tekstove iz širokog područja domena i namjene sa ciljem da što bolje predstavljaju cjelokupan jezik. Mogu sadržavati pisani i izgovoreni jezik te su relativno veliki, uglavnom sadrže više desetaka ili stotina milijuna riječi. Najčešće se koriste za izradu materijala za učenje jezika i prijevoda (riječnici, gramatike) ili kao bazna usporedba sa manjim specijaliziranim korpusima. Tipični primjer općejezičnog korpusa za engleski jezik je British National Corpus (BNC) koji sadrži oko 100 milijuna riječi.

Za razliku od općjezičnih korpusa, specijalizirani korpusi sadrže podskup jezika određene domene, primjerice medicinski ili akademski diskurs. Takvi korpusi su najčešće znatno manji od općjezičnih te mogu imati i druga ograničenja osim domenskih, kao što su žanr, vrijeme, geografska raznolikost itd.

Također važna podjela je između sinkronijskih i dijakronijskih korpusa. Sinkronijski opisuju stanje jezika u nekom kraćem vremenskom periodu dok dijakronijski korpusi prikazuju razvoj jezika kroz vrijeme. Posebni dijakronijski korpusi nazivaju se monitor ili dinamičkim korpusima jer se kontinuirano povečavaju. Ovakve korpuse je predložio Sinclair (1991) te teži da se napravi korpus koji stalno raste.

Prijevodni korpusi sadrže zbirku originalnih tekstova na jednom jeziku te njihove prijevode na jednom ili više drugih jezika, ukoliko su jednosmjerni, dvosmjerni korpusi sadržavaju originale i prijevode za svaki od jezika u prijevodnom korpusu. Usporedivi korpusi sadrže tekstove na dva ili više jezika ali nisu direktno prevedeni nego su iz iste domene, kategorije, područja ili tematike. Do sada navedeni korpusi su višejezični jer sadržavaju dva ili više jezika, točnije korpus je višejezičan ako sadrži tri ili više jezika dok je dvojezičan kada sadrži dva jezika.

## Web kao korpus

Ideja da se web tretira kao korpus zahtjeva da se ponovno definira što korpus jest ili što sve korpus može biti. Osnovna definicija korpusa kao skupa tekstova u jednoj cjelini se može shvatiti na način da je bilo koji skup od dva ili više tekstova korpus. Iako je web skup tekstova u obliku čitljivom za računalo nameće se problem da nema lingvističku namjenu niti je stvoren da bude korpus(Gatto 2:2). Pojavom namjere lingvista da koriste web kao korpus potaklo je ponovnu definiciju korpusa:

Želimo izbjeći *švercanje* vrijednosti u kriteriji korpusa. McEnery i Wilson mješaju pitanje „Što je korpus?“ sa pitanjem „Što je dobar korpus?“ (za određeno istraživanje), time zamućuju jednostavno pitanje „Je li korpus x dobar za istraživanje y?“ sa semantičkim pitanjem „Je li x uopće korpus?“. Takvo semantičko pitanje odvlači pažnju od odgovaranja na praktična pitanja. Da bi se ovakva semantička pitanja odstranila, definicija korpusa treba biti široka. Mi definiramo korpus kao „skup tekstova“. Definicija je općenita, također ovisi o kontekstu i domeni u kojoj se koristi, dakle preciznije: „Korpus je skup tekstova kada se promatra kao objekt jezika ili književnog istraživanja.“(Kilgarriff, Grafenstette: 2003 Computational Linguistics)

Istraživanja u domeni web-a kao korpusa dugo su bila ograničena na praktično pitanje da li web kao korpus može biti dobar za određenu zadaću, dok su u stvarnosti upravo ta istraživanja preoblikovala definiciju korpusa u novom mileniju (Gatto 2:2). Pokušavajući definirati korpus, razni priznati izvori pokazali su probleme definiranja prirode korpusa kao objekta istraživanja jezika. Izazovi koji se pojavljuju u pripremi i korištenju weba kao korpusa poslužili su kao „povečalo za metodološke probleme o s kojima se korpusni lingvisti oduvjek susreću“(Hundt et al. 2007: 4). Nova istraživanja mogu samo doprinjeti korpusnoj lingvistici, razvojem novih alata i metoda da bi se zadovoljile praktične potrebe istraživača i unaprijedile postojeće metode i resursi.

## 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). Kada se na web gleda kao na korpus onda je ovoj preporuci potrebna revizija. Što veće moguće za korpus koji se sastoji od weba znači nekontroliranu i neizmjernu količinu podataka. Ukoliko postoji razlog sa lingvističke strane, da korpus bude što veći mogu se pojaviti problemi u metodama prikupljanja i pohranjivanja podataka. Trenutna veličina indeksiranog weba je 4.8 bilijuna web stranica (<http://www.worldwidewebsize.com/> 18.1.2017) što još uvijek nije ni približno potpuna veličina weba, naročito iz perspektive korpusne lingvistike gdje se veličina korpusa mjeri brojem pojavnica.

## Važnost i uloga korpusa

Korpusi pružaju empirijske podatke koji lingvistima omogućavaju da postave objektivne tvrdnje utemeljene na istraživanju jezika onakvog kakav jest, za razliku od tradicionalnih subjektivnih tvrdnji koje se temelje na introspekciji pojedinca. Korpusi se koriste za morfološka i sintaktička istraživanja, poput istraživanja svih oblika određene riječi. Osim što sadrže jezik koji je bio ili je trenutno u upotrebi, također mogu sadržavati i podatke o autoru, žanru, regionalnoj varijaciji i vremenu nastanka teksta.

Podaci u korpusu mogu se podjeliti na tri vrste(Tadić 2003:31):

* evidencija – pronalaženjem ima li neke jezične jedinice u teksu
* frekvencija – ukoliko postoji jezična jedinica u tekstu koliko puta se pojavljuje
* relacija – pronalaženje u kakvom odnosu jezična jedinica stoji prema drugim jezičnim jedinicama

Specijalizirani korpusi sastavljeni od određenog žanra tekstova ili od tekstova određenog autora značajan su izvor podataka u stilističkim istraživanjima. Korpusi koji su označeni prema regionalnoj varijaciji služe u istraživanju dijalektologije. Forenzička lingvistika koristi korpuse pri analizi vjerodostojnosti dokumenata koji se koriste u pravnim procesima ili identifikaciji autorstva u akademskim okruženjima.

Višejezični usporedni korpusi značajni su pri učenju stranih jezika jer takvi korpusi mogu služiti kao riječnici prilagođeni učenju jezika. Ovakve korpuse također koriste prevodioci za usporedbu vlastitog prijevoda s već postojećim prijevodom ili originalom za koji postoji korpus.

Podjelu korpusne lingvistike koju uvodi Tognini-Bonelli koja istraživanja dijeli na ona koja su temeljena na korpusima (eng. *corpus based approach*) u smislu da istraživač postavi hipotezu pa točnost iste provjerava istraživanjem korpusa, i ona istraživanja koja su potaknuta korpusima (eng. *corpus driven approach*) u smislu da se sami korpus uzima kao prikaz jezične upotrebe pa se dolazi do zaključaka bez prethodno postavljene hipoteze, u takvom istraživanju korpus ne uzima kao dokaz ili osporavanje neke hipoteze već se zaključci donose na osnovu onoga što se nalazi u korpusu. U bilo kojoj od ove dvije vrste istraživanja korpusa važno je eksplicitno navesti parametre po kojima je sastavljen korpus te navesti sve detalje o tekstovima od kojih se korpus sastoji kako bi se moglo zaključiti o kakvoj se jazičnoj pojavi radi.

# 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 Toolkit uključuje ogledne primjere 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.

Tekstovi prikupljeni u poglavlju 2. Prikupljanje podataka služe za izradu korpusa koji predstavlja neformalnu komunikaciju na internetskom forumu i društvenoj mreži. Budući da je stvaranje korpusa koji bi predstavljao cjelokupnu komunikaciju preko interneta, daleko izvan dometa ovog rada, korpus koji je napravljen služi da bi se pokazale mogućnosti NLTK biblioteke. Korpus je izrađen od 163049 objava koje su objavljene između 2008. i 2017. godine na bug.hr forumu[link] te facebook grupama Sistemci Hrvatska, Developers Hrvatska i IT Jobs Croatia. Ukupni broj pojavnica je XX od čega jedinstvenih XX.

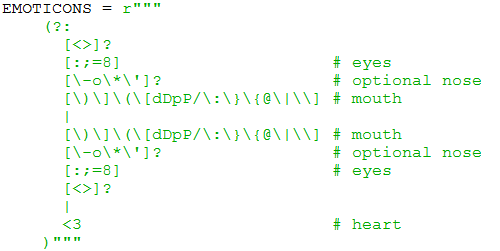
## Opojavničenje

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 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 točno odvojiti pojavnicu. Tekst koji se obrađuje može se opojavničiti više puta da bi se dobili željeni rezultati. Nakon dobivanja pojavnica koje su riječi ponekad je potrebno spojiti više riječi koje, gotovo uvijek, imaju isto značenje, primjerice „tvrdi disk“. 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 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 društvene mreže facebook i tweeter koriste 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. Natural Language Processing Toolkit sadrži više modula za opojavničenje teksta koji su najmenjeni određenom jeziku ili tipu teksta.

### Opojavničenje prikupljenih tekstova

Podaci prikupljeni u 2. Prikupljanje podataka sadrže pojavnice i znakove koji su tipični za internetske forume i društvene mreže. Natural Language Processing Toolkit (NLTK) sadrži modul *nltk.tokenize.casual* koji koristi listu regularnih izraza upravo za tekstove neformalnog komuniciranja na društvenim mrežama i sličnim platformama. Korištenjem regularnih izraza izdvajaju se pojavnice poput email adresa, emotikona i URL-ova. Pri instanciranju objekta *nltk.tokenize.casual* korisniku može dodati parametar *preserve\_case=False* što sve pojavnice osim emotikona pretvara u mala slova, zadana vrijednost ovog parametra je *True* što znači da će se zadržati velika slova u pojavnicama. Budući da se ovaj modul za opojavničenje bazira na regularnim izrazima lako ga je prilagoditi za tekstove sa specifičnim značajkama.



Slika 8 Regularni izraz za "emotikone"

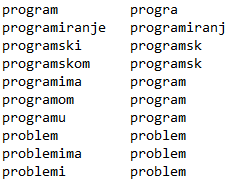
Osim modula koji opojavniče tekst gdje je pojavnica riječ, NLTK također sadrži modul *mwe* (eng. *Multi-Word Expression*) koji povezuje više riječi koje čine kolokaciju. Kolokacije su sintagmatske kombinacije dviju riječi koje čine sintagmu (Murica 2010:1). Prije upotrebe ove metode opojavničenja potrebno je istražiti koje pojavnice se pojavljuju zajedno. Funkcija *bigrams* generira listu parova pojavnica koje se pojavljuju jedna za drugom. Kolokacije u hrvatskom jeziku dostupne su za pretraživanje u Kolokacijskoj bazi hrvatskoga jezika [link]. Bigrami mogu biti kolokacije ako je pojavljivanje takvih bigrama učestalije u korpusu koji proučavamo. Funkcija *collocations* prikazuje ngrame po učestalosti pojavljivanja. Parametrima funkcije možemo odrediti broj pojavnica koji se spajaju u kolokaciju. Pri inicijaliziranju objekta koji predstavlja MWE tokenizer parametri su lista pojavnica koja se spaja, lista riječi koje čine izraz te znak kojim će se izrazi spajati.

Nakon opojavničenja teksta 163049 objava korisnika facebook društvene mreže i bug.hr foruma koristeći *casual* modul korpus se sastoji od XX pojavnica, od čega XX jedinstvenih pojavnica. Dijeljenjem broja jedinstvenih pojavnica sa ukupnim brojem pojavnica dobije se mjera leksičke raznolikosti teksta. Što je mjera manja, tj. bliža nuli u tekstu je prisutan manji broj različitih riječi a što je veća broj različitih riječi je veći.

## Korjenovanje

Nakon opojavničenja korpusa gdje su pojavnice riječi, korpus je potrebno morfološki normalizirati. Dva su osnovna pristupa morfološkoj normalizaciji: lematizacija i korjenovanje. Lematizacija je postupak kojim se pronalazi kanonski oblik riječi, tj. natuknički oblik u rječniku ili lema. Lematizator bi oblike *članstvo, članstva, članstava, članstvima* trebao svesti na lemu *članstvo*. Korjenovanje je pak postupak kojim se uklanjaju afiksi iz različitih oblika riječi kako bi našao „korijen” zajednički svim oblicima. Tako dobiven „korijen” češće je zapravo jednak osnovi riječi. Također razlikujemo slabo (konzervativno) korjenovanje koje obuhvaća samo flektivnu morfološku varijaciju, te jaku (agresivnu) koja obuhvaća i derivacijsku morfološku varijaciju. Slabo korjenovanje bi oblike *članstvo, članstva, članstava, članstvima* svelo na korijen *članstv* a postupak jakog korijenovanja bi trebao svesti na *član* (Pandžić 2015:1).

Za korjenovanje pojavnica u korpusu korišten je konzervativni korjenovatelj za hrvatski jezik od autora Nikola Ljubešić i Ivan Pandžić. Korjenovatelj sadrži afikse u obliku regularnih izraza koje se nalaze u datoteci *rules.txt* te prima pojavnice i pretvara ih u korjene. Cilj upotrebe korjenovatelja na ovom korpusu je smanjivanje broja pojavnica, tj. grupiranje pojavnica prema semantičkoj sličnosti. Kao i kod pravila za opojavničenje teksta, neformalan tj. nestandardizirani način komunikacije na društvenim mrežama i forumima otežava proces morfološke normalizacije te mu umanjuje preciznost i korisnost.



Slika 9 Korjenovanje pojavnica iz korpusa

# Klasifikacija teksta

Stalni rast količine tekstualnog sadržaja na webu stvara potrebu da se tekstovi razvrstaju, organiziraju i klasificiraju. Klasifikacija teksta je podjela teksta u kategorije koje su određene zadanim uvjetima. Tekst koji pišu korisnici, primjerice na internetskim forumima, često se klasificira po izboru korisnika tj. korisnik bira u kojoj kategoriji se tekst nalazi čak i prije pisanja. Kada se radi o večem broju tekstova kojima je potrebna klasifikacija, primjerice određivanje da li je elektronička pošta spam, ručna klasifikacija nije prikladno rješenje. Za ovakve i slične potrebe koristi se automatska klasifikacija. Jedan od popularnijih pristupa rješavanju ovog problema je korištenje tehnika strojnog učenja. Klasifikacija se može podijeliti na nadziranu i ne nadziranu. Klasifikacija je nadzirana ako klasifikatoru dajemo primjerak za treniranje koji je već klasificiran. Ne nadzirana klasifikacija nema klasificirani primjerak tako da algoritam grupira tekstove prema zajedničkim značajkama.

## Naivni Bayesov klasifikator

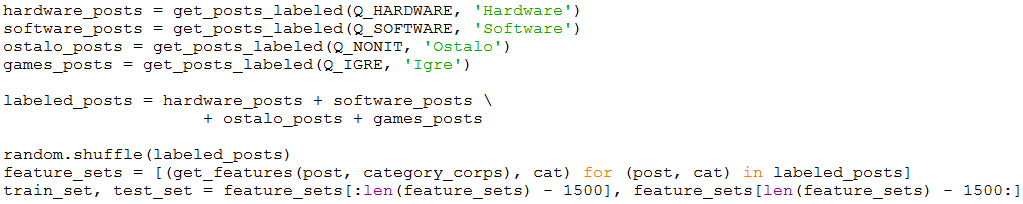
Naivni Bayesov klasifikator radi na principu Bayesovog teorema koji opisuje vjerojatnost događaja koja se temelji na poznavanju nekih uvjeta vezanih za taj događaj. Natural Language Processing Toolkit sadrži jednostavnu implementaciju naivnog Bayesovog klasifikatora koji radi na principu:

Algoritam radi „naivnu“ pretpostavku da su sve značajke nezavisne za danu kategoriju. Dakle, algoritam računa koja je vjerojatnost da kategorija ima određene značajke. Budući da je implementacija algoritma ugrađena u NLTK, jedino je potrebno odabrati koje značajke teksta najbolje određuju kategoriju teksta. Odabir relevantnih značajki i način na koji ih izražavamo može imati veliki utjecaj na uporabljivost klasifikatora. Iako je moguće dobiti dobre performanse klasifikacije koristeći jednostavne i očite značajke, značajna povečanja u preciznosti ostvaruju se značajkama koje zahtjevaju dobro poznavanje tipa tekstova koje klasificiramo. Značajke se često odabiru metodom pokušaj-pogreška jer intuitivni odabir nemora nužno biti koristan za klasifikaciju teksta tako da je potrebno provjeravati preciznost klasifikatora svakim uvođenjem novih ili izmjenom postojećih značajki.

Objave korisnika prikupljene u poglavlju 2. Prikupljanje podataka pokušati ćemo automatski klasificirati koristeći naivni Bayesov klasifikator. U bazi podataka prikupljena je 91 tema koja označava tematiku objave. Radi malog broja objava u nekim temama te sličnosti mnogih tema odabrane su 4 šire teme: Hardver, Softver, Igre i Ostalo.

|  |  |
| --- | --- |
| Tema | Broj objava |
| Hardver | 12826 |
| Softver | 14654 |
| Igre | 27075 |
| Ostalo | 9771 |

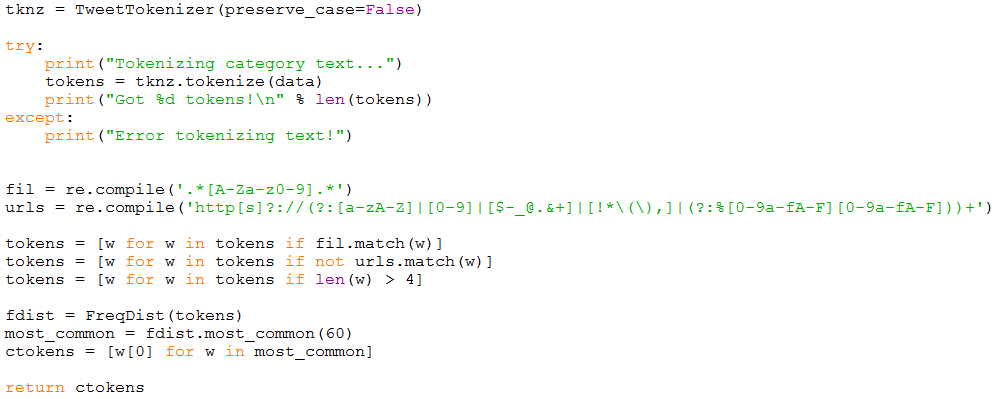
Klasifikator prvo treba „naučiti“ koja svojstva čine kategoriju. Ovo postižemo davanjem označenih podataka klasifikatoru koji nad tim podacima koristi funkciju za dobivanje svojstva. Ovim postupkom klasifikator određuje koja je vjerojatnost da neka kategorija ima određena svojstva. Jedan od ključnih dijelova izrade i testiranja klasifikatora je podjela podataka na dio za učenje i dio za testiranje.



Slika Setovi za učenje i testiranje

Dohvaćaju se sve objave iz baze podataka te se označuju sa odgovarajućom kategorijom. Nakon spajanja objava koristi se funkcija *random.shuffle* da bi se podaci izmješali tj. da klasifikator dobije podatke za svaku kategoriju. Varijable *feature\_sets* sadrži svojstva i kategoriju objava, odvajanjem podataka tako da je znatno veći broj dodjeljen varijabli *train\_set* koja „uči“ klasifikator.

Najvažniji dio u izradi klasifikatora je odabir svojstava. Za ovaj klasifikator odabrana su svojstva definirana korištenjem korpusnih metoda opisanih u 4.2 i 4.3. Pod pretpostavkom da svaka od kategorija sadrži neke pojavnice koje su specifične za tu kategoriju i večinom se pojavljuju samo u njoj, jedno od svojstva je brojanje pojavljivanja pojavnica iz najčešćih pojavnica neke kategorije. Da bi se izbjegle pojavnice koje su česte općenito u hrvatskom jeziku, primjerice veznici, izrada riječnika čestih pojavnica kategorije odnosi se na pojavnice sa više od 4 slova.



Slika Česte pojavnice kategorije

Za svaku kategoriju se izrađuje ovakav skup pojavnica. Skupovi se koriste da bi se odredilo svojstvo koje izražavamo brojem pojavnica iz riječnika pojavnica te kategorije. Osim ovog svojstva koristi se jednostavniji pristup a to je broj riječi u objavi.

Iako se odabrana svojstva čine intuitivna, rezultati klasifikacije ukazuju na teškoće u diferenciranju kategorija. Točnost klasifikatora na 1500 testnih objava je 42.33% dok je najinformativnije svojstvo broj riječi u objavi. Za bolje rezultate potrebno je dodatno rafinirati svojstva koja su odabrana, primjerice često korištene riječi u svim kategorijama su: hvala, pozdrav, netko, zanima, nakon, ovako. Ručnim odabirom pojavnica određene kategorije te morfološkom normalizacijom tekstova moguće je poboljšanje klasifikatora.

# Zaključak

# Dodaci

## Razlike između Python 2.x i Python 3.x

Python 3 izašao je u prosincu 2008. godine dok je zadnja inačica 2.x verzije bila 2.7 koja je izašla u srpnju 2010. godine. Python 3 je u aktivnom razvoju, za vrijeme pisanja ovog rada zadnja inačica je 3.6.0 a prema rasporedu 3.6.1 treba izaći u ožujku 2017. godine.

Guido van Rossum (autor programskog jezika Python), odlučuje da se Python 2.x treba poboljšati te da neke od promijena koje su nužne neće biti backward compatible sa starijim inačicama. Najdrastičnije promijene se tiču Unicode podrške, konkretnije svi znakovni nizovi (eng. strings) su po zadanoj vrijednosti Unicode tipa. Tjekom pisanja ovog rada autor se susretao sa problemima odabira inačice Python programskog jezika, te korištenja modula koji su nekad bili pisani u starijoj (2.x inačici) a nekad u novijoj. Službeni stav Python zajednice je da se za nove projekte koristi Python 3.x ako ne zahtjeva module i biblioteke koje (još) nisu prebačene u najnoviju inačicu. Potencijalni problem korištenja najnovije Python inačice je nedostatak dokumentacije ili dokumentacija koja se odnosi na stariji Python 2.7. Ako se radi o nadogradnji ili održavanju postojećih programa koji su napisani u 2.x verziji, prebacivanje na noviju verziju može biti teško, naročito kod programa koji sadržavaju velike količine programskog koda.

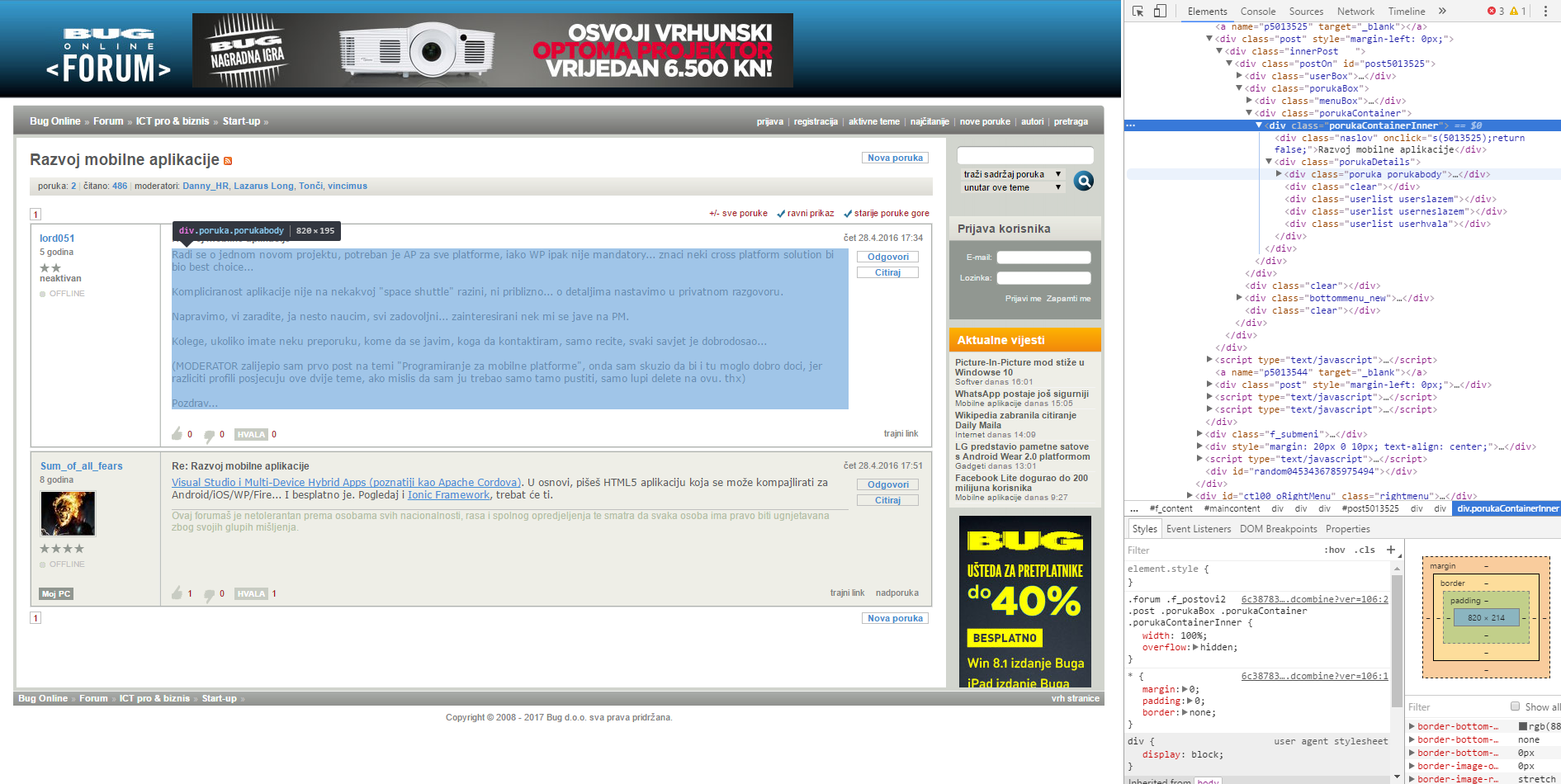
Moduli i biblioteke koje se koriste u velikom broju programskih rješenja u večini su napravile tranziciju na Python:

* NumPy i SciPy – moduli za znanstvena istraživanja
* Django, Flask i Pyramid – razvojne cjeline za Web
* cx\_Freeze – skupina skripta za distribuciju i menađment dependency-ja
* py2exe – distribucija programa za Windows platformu
* OpenCV 3 – biblioteka za obradu grafičkog sadržaja i strojno učenje
* Requests – HTTP biblioteka
* lxml – biblioteka za manipulaciju XML datotekama
* BeautifulSoup – biblioteka za manipulaciju i parsiranje XML i HTML datoteka

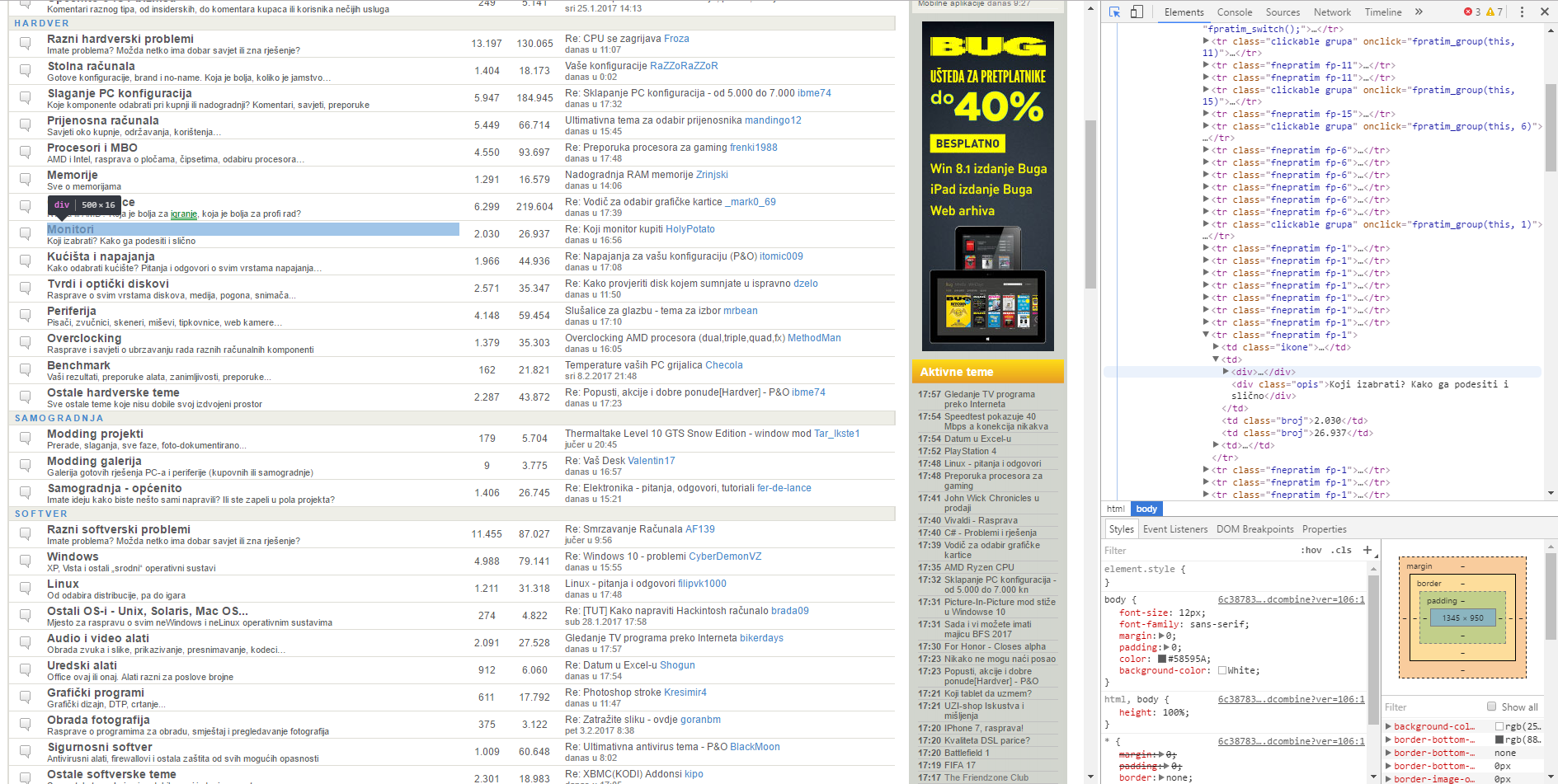
Službena dokumentacija također pruža naputke za tranziciju u Python 3 te standardna biblioteka sadrži pomoćne programe kao što su *2to3* i *3to2* koji automatiziraju prijetvorbu programskog koda[Python dokumentacija].

U ovome radu korištene su obje inačice programskog jezika Python. U poglavlju 2. Prikupljanje podataka programski kod je napisan u 2.7 inačici jer je dokumentacija modula i API-ja koji se koristi uglavnom napisana za verziju 2.7. U poglavlju 4. Izrada korpusa koristi se Python 3.6 upravo radi bolje podrške za Unicode i nekih značajki modula NLTK koje su dostupne samo u verziji koja radi na Python-u 3.[Službena doku. Python]

## Bug forum HTML struktura



Slika HTML struktura objave



Slika HTML struktura teme

# Literatura

Tony McEnery, Andrew Hardie: „Corpus Linguistics“, Cambridge: Cambridge University

Press 2012

Maristella Gatto: „Web as Corpus“, 2014

Tognini-Bonelli. E. (2001).. Corpus Linguistics at Work. Amsterdam: J. Benjamins

Zipf, G.K. (1935), „The psychobiology of language“. New York: Houghton Mifflin

Kilgarriff, A., Grefenstette, G. (2003), „Introduction to the Special Issue on the Web as a Corpus“, Computational Linguistics

Tadić. M. (2003). Jezične tehnologije i hrvatski jezik. Zagreb: Ex libris

Pandžić. I. (2015), Oblikovanje korjenovatelja za hrvatski jezik. Časopis Instituta za hrvatski jezik i jezikoslovlje 41/2