## Prikupljanje podataka

World wide web je moguće predstaviti kao veoma veliki usmereni graf u kome web stranice predstavljaju čvorove, a linkovi između stranica predstavljaju grane grafa.

World wide web svakako predstavlja najznačajniji izvor podataka. Podaci na webu se najčešće pronalaze u formi web stranica, koje za cilj imaju da podatke predstave korisniku na način koji će za čoveka biti intuitivan. Ipak kada je u pitanju rad sa velikom količinom podataka koju prikuplja i obrađuje računar, ovaj način predstavljanja podataka nije naročito efikasan. Ipak, kako struktuirani grupisani podaci nisu uvek dostupni za prikupljanje, često dolazi do potrebe za prikupljanjem podataka sa web stranica. Jedna od najčešćih tehnika u prikupljanju podataka sa web stranica jeste korišćenje web crawler komponenti. Web crawler je softverska komponenta koja koristi grafovsku strukturu world wide weba u cilju omogućavanja pretrage međusobno povezanih stranica bez ljudske interakcije. Suština web crawler komponenti jeste pretraga html sadržaja web stranica iz kojeg je moguće prikupiti tražene podatke, kao i linkove ka daljim stanicama od interesa. Kako bi pretraga web stranica korišćenjem crawlera bila optimizovana, uvek je poželjno da struktura sajta koji se pretražuje bude smislena kao da je i html struktura samih stranica pogodna za web crawling. Ove osobine su veoma značajne i kada je u pitanju rangiranje web sajtova na raznim sistemima pretrage, kao što su Google ili Bing.

Web crawler komponenta ima za cilj pristupanje skupu međusobno povezanih stranica korišćenjem nekog od algoritama za obilazak grafova, i prikupljanje značajnih podataka tokom njihovog obilaska. Kada je u pitanju obilazak željenih stranica, prvo što je potrebno odrediti jeste skup početnih stranica obilaska. Prilikom ovog koraka je najvažnije pažljivo odabrati stranice iz kojih se putem linkova može stići do što većeg broja drugih stranica od značaja (Slika x.1). Ukoliko se ovaj korak ne izvrši na dobar način, čitav proces prikupljanja podataka može biti neuspešan.

//////////////////////////////////////////////////SLIKA X1///////////////////////////////////////////////////

Web crawler komponenta mora sadržati kolekciju linkova ka stranicama koje je potrebno posetiti, kao i kolekciju stranica koje su već posećene kako bi se izbeglo zaglavljivanje web crawlera u slučaju pojave konture u grafu (Slika x.2). Prilikom posete svakoj od stranica, link posećene stranice se najpre prebacuje iz skupa neposećenih u skup posećenih stranica, potom se sa stranice ekstraktuju svi potrebni podaci. Poslednji korak koji se obavlja na svakoj stranici je izvlačenje linkova ka ostalim stranicama, i njihovo smeštanje u kolekciju neposećenih ukoliko stranice na tim linkovima već nisu posećene. Kraj procesa prolaska kroz sajt nastupa u trenutku kada kolekcija neposećenih linkova više ne sarži ni jedan element.

//////////////////////////////////////////////////SLIKA X2///////////////////////////////////////////////////

Kako bi web crawler komponenta imala smisao, potrebno je implementirati segment za analizu html sadržaja web stranice, kao i parsiranje podataka sa iste. Prilikom posete svake od stranica komponenta zadužena za prikupljanje podataka izvršava tri koraka: pristupanje traženim html elementima, parsiranje vrednosti elemenata u podatke i struktuiranje i čuvanje podataka. Kada su u pitanju prva dva koraka, oni uveliko zavise od same html strukture stranice, što je veoma često najveći izazov u implementaciji komponenti za prikupljanje podataka. Kako bi pravilno prikupljanje podataka sa web stranice bilo moguće, sama stranica mora imati jasno definisanu strukturu. Takođe kako bi se omogućilo lakše preuzimanje podataka, web stranice često koriste identifikacione html atribute kao što su id ili data atributi, pomoću kojih web crawler komponente mogu direktno pristupiti željenim html elementima i njihovim vrednostima. Nakon prikupljanja traženih podataka sa stranice, poslednji korak jeste struktuiranje i čuvanje istih. Ovaj proces počinje dizajnom strukture podataka, koja zavisi od samog cilja istraživanja. Ove strukture podataka često se nazivaju i modeli. Modeli u svojoj strukturi sadrže atribute čije vrednosti bivaju određene na osnovu prikupljenih podataka iz prethodnih koraka. Prilikom kreiranja modela, potrebno je dodeliti vrednost svim atributima koji su dostupni i neophodni za dalje istraživanje. Nakon što su novi podaci prikupljeni i struktuirani u modele, najčešća praksa jeste njihovo skladištenje, kako bi se omogućio kasniji pristup bez potrebe za ponovnom analizom web sadržaja.

Kao što je moguće videti iz prethodnih nekoliko pasusa, upotreba web crawler komponenti je veoma efikasan način pravilnog prikupljanja podataka koji se nalaze u formatu web stranica, te njihovog struktuiranja i skladištenja za kasniju upotrebu.

Početna tačka ovog istraživanja jeste prikupljanje podataka o naučnim radovima. Za potrebe ovog rada podaci su prikupljani iz arhive časopisa Computer Science and Information Systems (ComSIS) [Ref]. ComSIS je naučni časopis otvorenog tipa koji objavljuje originalne naučne radove kako teorijskog, tako i komercijalnog odnosno industrijskog tipa, na temu računarskih nauka. Kako ovaj ovaj časopis nema ne nudi otvoren pristup skladištu podataka u vidu aplikacionog programskog interfejsa (API), jedini način pristupa podacima o naučnim radovima objavljenim u ovom časopisu jeste korišćenje web crawler komponente. Kako bi ovu komponentu bilo moguće implementirati najpre je potrebno utvrditi samu strukturu web sajta, kao i utvrditi skup početnih stranica sa kojih se pretraga može izvršavati. Nakon kraće analize ComSIS web sajta, moguće je uvideti jasnu i pravilnu strukturu linkovanja prikazanu na slici x.1. Nakon osvrta na strukturu sajta nameće se jasan zaključak da je najbolja početna tačka za komponentu pretraživanja stranica arhive časopisa, jer je sa te stranice putem linkova moguće pristupiti svim ostalim stranicama od interesa.

Sama struktura web crawler komponente prikazana je na slici x.2. Kao što je moguće videti na dijagramu kalasa sa slike, konstruisan je interfejs PublicationsWebCrawler namenjen za korišćenje od strane eksternih komponenti. Klasa *ComSisPublicationWebCrawler* predstavlja implementaciju ovog interfejsa, i kako je propisano interfejsom sadrži samo jedan metod *searchForPublications* koja za cilj ima inicijalizaciju procesa za pretragu i njihovo upravljanje. Prilikom vršenja ovog istraživanja klasa *ComSisPublicationWebCrawler* konfigurisana je da inicijalizuje deset procesa pretraživanja koji se paralelno izvršavaju. Prilikom inicijalizacije ova klasa kreira instance skupa posećenih stranica, skupa parsiranih publikacija kao i red opsluživanja stranica koje tek trebaju biti posećene. Ova tri polja su veoma važa zbog toga što se prosleđuju svim procesima pretraživanja koji ih potom koriste za skladištenje izvučenih podataka i određivanje završetka pretrage. S obzirom na činjenicu da su ova tri polja deljena na nivou više procesa, veoma je važno da strukture podataka koje se koriste budu sinhronizovane, odnosno da se onemogući pristup strukturi od strane više procesa istovremeno. Klasa *ComSisPublicationWebCrawler* startuje deset instanci klase *ComSisWebCrawlerThread* i pokreće ih svake sekunde. Razlog zbog koga se nakon svake iteracije proces zaustavlja na jednu sekundu je zbog usporavanja slanja http zahteva. Ovaj korak je veoma važan jer u slučaju kreiranja prevelikog broja zahteva u kratkom vremenskom periodu, često dolazi do preopterećenja servera. Takođe mnoge web aplikacije poseduju zaštitu od ovakvog tipa „napada“, što rezultuje blokiranjem zahteva sa određene IP adrese. Takođe nakon svake od iteracija potrebno je proveriti da li je proces pretrage završen. Ovaj podatak se računa tako što program iterira kroz sve instance klase *ComSisWebCrawlerThread* i proverava da li je pretraga za njih završena. Ukoliko su svi procesi pretrage završeni, skup prikupljenih modela vraća se korisniku. Implementacija ovog procesa prikazana je na listingu x.3



Centralna klasa web crawler komponente jeste *ComSisWebCrawlerThread.* Ova klasa nasleđuje je naslednica klase *java.lang.Thread*, što omogućava objektima ove klase da izvršavaju kod paralelno sa drugim procesima istog ili drugog tipa. Ovakav pristup je veoma pogodan kada je u pitanju pretraživanje web sajtova koji često mogu biti veoma obimni, stoga se korišćenjem paralelne obrade stranica značajno skraćuje vreme pretraživanja sajta. Klasa *ComSisWebCrawlerThread*  sadrži *run* metod koji pokreće proces učitavanja web stranica i njhovo analiziranje i parsiranje. Prilikom poziva ovog metoda proces ulazi u beskonačnu petlju i na taj način omogućava da se proces izvršava dokle god ne bude prekinut od strane klase koja ga je pokrenula. U svakoj iteraciji ove petlje objekat najpre proverava red opsluživanja koji sadrži objekte klase *WebPageData*. Svaki od ovih objekata sadrži link ka stranici koja do sada nije posećena, kao i nivo na kojoj se ta stranica nalazi. Ukoliko ovaj red nije prazan sa njega će biti preuzet prvi objekat klase *WebPageData*, i na osnovu njegovog linka biće upućen zahtev za preuzimanje html sadržaja web stranice. Za potrebe ovog istraživanja preuzimanje html sadržaja vršeno je pomoću javine biblioteke Jsoup. Glavna klasa ove biblioteke takođe sadrži naziv *Jsoup* ona omogućava preuzimanje html sadržaja u formi objekta clase *Document*, koja je takođe deo ove biblioteke. Poslednji korak svake iteracije jeste procesiranje dokumenta, koje izvršava klasa *DocumentProcessor* na osnovu prosleđenog dokumenta kao i nivoa na kome se nalazi stranica koja je učitana. Kako bi se omogućilo prekidanje procesa nakon završetka pretrage, *ComSisWebCrawlerThread* sadrži metod isWaitingForUrls, koji vraća vrednost true ukoliko proces pokušava da dobavi sledeći objekat klase *WebPageData* iz reda opsluživanja.

*DocumentProcessor* je klasa zadužena za procesiranje html dokumenata. Ova klasa sadrži statički metod *processDocument* koji kao ulazne parametre prima html dokument, kao i nivo na kome se taj dokument nalazi. Na osnovu prosleđenog nivoa procesor odlučuje o načinu parsiranja dokumenta. Ilustracija ovog procesa odlučivanja prikazana je na slici x.x. Kao što je moguće videti na slici klasa *DocumentProcessor* sadrži tri metoda za parsiranje html dokumenta. Metod *loadIssues* koristi se prilikom obrade web stranice arhive časopisa. Ovaj metod sa html dokumenata prikuplja sve linkove koji odgovaraju formatu linkova ka brojevima časopisa, i potom na osnovu njih kreira nove instance klase *WebPageData* koje se potom smeštaju u red linkova koji još nisu posećeni. Metod *loadPublications* poziva se kada je dubina na kojoj se stranica nalazi jednaka broju jedan. Ovaj metod se izvršava skoro potpuno isto kao i prethodni. Prilikom pozivanja ovog metoda, sa stranice se parsiraju svi linkovi koji odgovaraju formatu linkova ka stranicama sa naučnim radovima, i na osnovu njih se kao i u prošlom slučaju kreiraju novi objekti tipa *WebPageData* koji se smeštaju u red opsluživanja. Kada je dubina stranice jednaka 2, unutar klase *DocumentProcessor* poziva se metod *extractPublicationData.* Prilikom poziva ovog metoda prvi korak koji se izvršava jeste pronalaženje vrednosti unutar html elemenata na osnovu unapred dodeljenih selektora, nastalim analizom html sadržaja stranica na kojima se nalaze publikacije. Sledeći korak koji se izvršava u okviru ovog metoda jeste kreiranje objekta klase *PublicationPage* na osnovu prikupljenih atributa, nakon čega se kreirani objekat smešta u kolekciju prikupljenih modela deljenu na nivou svih *ComSisWebCrawlerThread*  procesa.

Nakon završetka pretraživanja web stranica svi web crawler procesi bivaju prekinuti od strane objekta klase *ComSisPublicationWebCrawler.* Poslednji korak glavnog procesa jeste vraćanje kolekcije sa objektima klase *PublicationPage,* popunjene modelima prikupljenih sa web stranica. Klasa *PublicationPage* sadrži neprilagođene podatke o naučnom radu, koji se moraju prilagoditi ili dopuniti kako bi bili pogodni za kasnije čuvanje i indeksiranje.

Nakon što klasa *IrPublicationService* od web crawler komponente dobije skup objekata *PublicationPage*, svaki se dodatno obrađuje i pretvara u instancu klase *PublicationData.* Glavni razlog za ovu konverziju jeste činjenica da se podaci sakupljeni prilikom procesa pretrage ne nalaze u formatu koji je pogodan za skladištenje. Najbolji primer ovoga jeste sekcija autora kao i sekcija sa afilijacijama i email adresama autora. Kao što je moguće videti na slici x.x, spisak autora je odvojen od sekcije sa njihovim afilijacijama, te je prostom selekcijom html elemenata stranice nemoguće povezati autore sa odovarajućom afilijacijom. Kako bi se ovaj problem prevazišao korišćena je klasa *AuthorsPageDataParser* koja najpre otklanja ostatke html tagova iz samih vrednosti nakon čega na osnovu spiska autora i njihovih referenci, za svakog autora kreira servisni model koji sadrži podatke o imenu autora, njegovoj email adresi i njegovoj afilijaciji. Drugi proces dopunjavanja podataka jeste konstruisanje url-a za preuzimanje dokumenta kombinovanjem adrese sajta dobijene iz konfiguracije i relativne putanje za preuzimanje dobijene sa stranici prilikom prikupljanja podataka.

Nakon što su svi podaci uspešno prikupljeni i parsirani sledi njihovo čuvanje u bazu podataka. Prilikom implementacije ovog projekta korišćena je PosgreSQL baza podataka. Glavni razlog korišćenja baze podataka umesto direktnog indeksiranja jeste mogućnost dobavljanja podataka iz više različitih izvora pre samog procesa indeksiranja, kao i skladištenje podataka u slučaju potrebe za ponovnim indeksiranjem podataka.