Laboratorij za analizu teksta i inženjerstvo znanja – TakeLab

Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikrolektroniku, računalne i inteligentne sustave
Unska 3, 10000 Zagreb, Hrvatska

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1922

POSTUPAK AUTOMATSKOG IZLUČIVANJA TEKSTNOG SADRŽAJA IZ INTERNETSKIH STRANICA

Josip Bakić

Zagreb, rujan 2012.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 9. svibnja 2012.

Zavod: **Fakultet elektrotehnike i računarstva**
Predmet: **Izrada diplomskog rada**

DIPLOMSKI ZADATAK br. 1922

Pristupnik: **Josip Bakić**
Studij: Računarstvo

Zadatak: **Postupak automatskog izlučivanja tekstnog sadržaja iz internetskih stranica**

Opis zadatka:

Na Internetu je dostupna velika količina tekstnih informacija. Glavni tekstni sadržaj internetskih stranica uklopljen je u često vrlo složenu HTML-strukturu stranica. Izlučivanje čistoga tekstnog sadržaja iz HTML-dokumenata (engl. web page cleaning) nužan je korak predobrade kako bi se prikupljeni tekstovi mogli upotrijebiti u daljnjim postupcima dohvata informacija i dubinske analize teksta. Neki od suvremenih postupaka izlučivanja tekstnog sadržaja temelje se na heuristički utvrđenim pravilima dok se drugi temelje na metodama strojnog učenja.
U okviru završnog rada potrebno je proučiti postojeće postupke za izlučivanje tekstnog sadržaja iz HTML-dokumenata. Potrebno je razraditi postupak koji bi, koristeći metode nadziranog strojnog učenja, kombinirao postojeće pristupe, uvažavajući njihove individualne prednosti i nedostatke. Provesti temeljito vrednovanje razvijenog postupka te provesti analizu značajki i pogrešaka. Načiniti programsku izvedbu alata za izlučivanje tekstnog sadržaja u programskom jeziku po izboru. Alat dodatno treba podržavati definiranje uzoraka strukture HTML-dokumenata unutar kojih se nalazi glavni tekstni sadržaj. Na temelju korisnički definiranih uzoraka potrebno je izlučivati tekstni sadržaj HTML-stranica za koje prethodno implementirani automatski postupak ne daje zadovoljavajuće rezultate. Radu priložiti izvorni programski kod, programsku dokumentaciju i korištene skupove podataka.

Zadatak uručen pristupniku: 10. svibnja 2012.
Rok za predaju rada: 10. rujna 2012.

Mentor:

Doc.dr.sc. Jan Šnajder

Djelovođa:

Doc.dr.sc. Leonardo Jelenković

Predsjednik povjerenstva za diplomski ispit:

Prof.dr.sc. Marijan Đurek

Sadržaj

Popis tablica

1. Uvod

S naglim razvojem i porastom popularnosti interneta, a posebice World Wide Weba (u daljnjem tekstu: web) kao jedne od usluga koja se na internetu pruža, nalazimo se u situaciji da na internetu možemo pronaći velike količine relativno lako dostupnih tekstnih podataka. Ta je količina podataka neizbježno privukla pažnju brojnih stručnjaka i našla vrlo korisne primjene u leksikografiji, prevođenju, obradi prirodnog jezika (engl. *natural language processing*, NLP), dubinskoj analizi teksta i drugim područjima. Razvijeno je pregršt alata koji omogućuju iskorištavanje ovog podatkovnog blaga, i napisano mnogo stručnih članaka koji se bave webom kao izvorom podataka.

Glavna poteškoća u korištenju weba kao korpusa jest činjenica da web-stranice ne čini samo goli tekst. Format dokumenata na webu, HTML (engl. *HyperText Markup Language*), sadrži mnogo popratnog sadržaja – naredbe kojima se definira kako će sam tekst biti prezentiran čitatelju, meta-podatke, programski kod i sl. – koji nam nije zanimljiv. Također, uobičajeno je da se na svakoj web-stranici pored samog korisnog, glavnog teksta nalaze i razni drugi sadržaji. Primjerice, to su oglasi, izbornici, pa zaglavlja i podnožja koja su tipično zajednička svim stranicama unutar nekog portala i nevezana za sam tekst te razni drugi sadržaji od kojih su neki nevidljivi i krajnjem korisniku – čitatelju.

Velika prednost weba njegova je veličina. Po podacima od Googlea iz 2008. godine, na webu postoji najmanje bilijun (10^{12}) jedinstvenih web-adresa (engl. *Uniform Resource Locator*, URL, ili *Uniform Resource Identifier*, URI), a svakodnevno nastaje oko milijardu (10^{9}) novih web-stranica (Alpert i Hajaj, 2008).

Usporedimo li to s tradicionalnim tekstnim korpusima, tolika količina podataka koja je dostupna na webu osim veličine same po sebi ima i tu bitnu prednost da se za rijetke riječi i fraze, koje u nekom korpusu mogu biti jedva, ako i uopće, zastupljene, na webu tipično može pronaći pregršt primjera (Fletcher, 2011).

Činjenica da toliko novog sadržaja na webu nastaje svakodnevno daje webu još jednu veliku prednost nad tradicionalnim korpusima – svježinu. Novonastale riječi ili novi načini upotrebe riječi, koji u nekom tradicionalnom korpusu ne postoje jer su nastali nakon njegovog dovršetka, na webu su dostupni u velikom broju.

Kao govornicima jednog slabo rasprostranjenog jezika, hrvatskog, i to u zemlji u kojoj su ulaganja u znanost na nezavidnom nivou, web predstavlja dragocjenu

priliku da i uz skromnija financijska sredstva stvorimo tekstne korpuse dovoljne kvalitete i veličine da nam budu korisni u istraživanjima.

No, za razliku od tradicionalnih tekstnih korpusa, web je vrlo neuredan izvor podataka. Teško je, često praktički nemoguće, utvrditi izvor nekog teksta, njegova lingvistička svojstva, ili njegovu vjerodostojnost. Teško je utvrditi je li autoru teksta jezik u kojem je tekst napisan materinji tj. je li tekst lingvistički reprezentativan primjer upotrebe tog jezika. Radi li se možda o prijevodu? Radi li se o prijevodu stručnjaka ili možda niže kvalitetnom strojnom prijevodu?

Tu je i sivo područje autorskih prava. Iako je sav sadržaj vrlo lako javno dostupan, ne podrazumijeva se da je preuzimanje sadržaja dozvoljeno za upotrebu u druge svrhe ili daljnju distribuciju. Pravni okviri u raznim zemljama po ovom se pitanju značajno razlikuju, što je također problem. Postoje web-stranice koje svoj sadržaj nude pod neograničavajućim licencijama, no izdvajanje samo takvih stranica može biti mukotrpan dodatan posao, i može vrlo značajno ograničiti dostupne podatke. No, u pravilu, iako pravni rizik postoji, on je zanemariv.

Također, iz sakupljenih web-stranica (a već i sam proces sakupljanja, engl. *crawling*, iako jedno već vrlo dobro istraženo područje, ima svoje izazove) potrebno je izvući koristan tekstni sadržaj.

Postoje mnogi razvijeni algoritmi za rješavanje problema ekstrakcije teksnog sadržaja (engl. *content extraction*). U ovom radu pokušava se ostvariti hibridno rješenje, takvo koje bi uz pomoć metoda strojnog učenja kombiniralo nekoliko postojećih algoritama za ekstrakciju tekstnih sadržaja u cilju ostvarivanja boljeg prosječnog rezultata na većem skupu dokumenata od primjene tih algoritama pojedinačno na cijeli skup. U radu su predstavljene dvije iskušane metode kombiniranja, te dani rezultati svake i uspoređeni sa samim algoritmima koje kombiniraju. Također je izveden i ekstraktor tekstnog sadržaja koji vrši ekstrakciju na temelju ručno napisanih predložaka.

U drugom poglavlju obrađeni su načini na koje se vrednuju rezultati postupaka ekstrakcije tekstnog sadržaja. Zatim su u trećem poglavlju navedene metode ekstrakcije sadržaja koje su objavljene u stručnoj literaturi, među njima i neke koje će se koristiti u ovom radu. U četvtom poglavlju dan je opis postupka. U petom poglavlju navedeni su postignuti rezultati i uspoređeni su sa nekim referentnim metodama. U šestom poglavlju nalazi se opis programske izvedbe rješenja te na kraju slijedi zaključak.

2. Vrednovanje rezultata ekstrakcije

Nakon izvršene ekstrakcije teksta iz web-stranice, potrebno je ocijeniti uspješnost odabrane metode. Za to se mogu koriste tri uobičajene mjere: preciznost, odaziv i ocjena F1 (Gottron, 2007). U definiranju ovih mjera korisni su sljedeći pojmovi:

1) TP – *True Positive* – definiramo kao ukupan broj riječi u tekstu koji je metoda odabrala kao bitan, i koji to zbilja jest;
2) FP – *False Positive* – ukupan broj riječi u tekstu koji je metoda odabrala kao bitan, ali koji to nije;
3) TN – *True Negative* – ukupan broj riječi u tekstu koji je metoda ocijenila nebitnim, i koji to zbilja jest;
4) FN – *False Negative* – ukupan broj riječi u tekstu koji je metoda ocijenila nebitnim, ali koji je u stvari bitan.

Bitno je pitanje kako prirediti rezultat ekstrakcije i referentni, ručno očišćeni tekst, kako bi usporedbe između dvaju tekstova bile neosjetljive na nevažne razlike u oblikovanju praznina (engl. *whitespace*, ne uključuje samo oznaku razmaka između riječi, već i prijelaske u novi red, tabulatore i sl.). Jednostavan način jest da se tekst pretvori u niz riječi (Kovačić, 2011). Tekst se dijeli u niz riječi korištenjem praznine kao mjesta razdvajanja dvaju riječi.

Nakon pripreme teksta, potrebno je pronaći presjek ekstrahiranog teksta i "zlatnog standarda" (engl. *gold standard*, unaprijed pripremljeni, očišćeni tekst sa stranice koji služi kao referentan u vrednovanju rezultata). Za pronalaženje te duljine može se upotrijebiti jedan od algoritama za traženje najduljeg zajedničkog podniza. Najdulji zajednički podniz (engl. *longest common subsequence*) dvaju nizova riječi je opet niz riječi u kojem se nalaze sve riječi koje se nalaze u oba početna niza, u istom redoslijedu, ali ne nužno u jednom kontinuiranom bloku. Standardni algoritam iz obitelji algoritama dinamičkog programiranja koji nalazi duljinu najduljeg zajedničkog podniza dvaju nizova opisan je u (Hirschberg, 1975). Često se koristi i algoritam Ratcliff-Obershelp (Ratcliff i Metzener, 1988), koji istom problemu pristupa na nešto drugačiji način. Primjena ovog algoritma za svrhu vrednovanja rezultata ekstrakcije opisana je u (Krišto, 2009).

Nije potrebno odrediti točno o kojem se podnizu radi, dovoljna je informacija o njezinoj duljini. Duljina podniza između zlatnog standarda i ekstrahiranog teksta čini

TP. FP je određen kao razlika između TP i duljine ukupnog ekstrahiranog teksta, dok je FN jednak razlici između TP i ukupne duljine zlatnog standarda. TN za ocjenjivanje nije potreban. Sve veličine bi bile izražene u broju riječi.

Preciznost (engl. *precision*) neke metode definira se kao omjer TP i zbroja TP i FP, tj. jednaka je duljini najduljeg zajedničkog podniza podijeljenoj s ukupnom duljinom ekstrahiranog teksta.

$$p = \frac{TP}{TP + FP}$$

Odaziv (engl. *recall*) definira se kao omjer TP i zbroja TP i FN, tj. jednak je omjeru duljine najduljeg zajedničkog podniza i ukupne duljine zlatnog standarda.

$$r = \frac{TP}{TP + FN}$$

F1 je kombinirana ocjena izvedena iz vrijednosti preciznosti i odaziva, koja omogućuje lakše uspoređivanje pojedinih metoda jer umjesto dviju imamo samo jednu vrijednost. Definirana je kao harmonijska sredina preciznosti i odaziva.

$$F_1 = \frac{2 \cdot p \cdot r}{p + r}$$

Zanimljiva i često korištena alternativa gornjim trima mjerama je Levenshteinova udaljenost uređivanja (Levenshtein, 1966). Mjera vuče svoje porijeklo iz teorije informacije i definira se pomoću jediničnih operacija uređivanja: umetanja, brisanja, i zamjene jednog znaka u tekstu. Levenshteinova udaljenost uređivanja dvaju tekstova jest najmanji broj jediničnih operacija uređivanja koji je potreban da bi se iz jednog teksta dobio drugi.

Za potrebe vrednovanja ekstrakcije tekstnog sadržaja Levenshteinova mjera se može prilagoditi tako da umjesto nad pojedinim znakovima teksta radi nad cijelim riječima, iz istog razloga kao i prije – zbog eliminiranja osjetljivosti na nebitne razlike u prazninama. Oba se teksta pretvore u niz riječi, pa se traži Levenshteinova udaljenost između takva dva niza, a na način da se pojedinačna umetanja, brisanja i zamjene rade nad cijelim riječima odjednom.

3. Metode ekstrakcije teksta iz web-stranica

Ekstrakcija teksta iz web-stranica, odnosno ekstrakcija sadržaja (engl. *content extraction*), aktivno je i dobro istraženo područje. Postoje razne metode o kojima su napisani mnogi radovi. Ovdje je dan pregled nekih zanimljivih, karakterističnih pristupa ovom problemu koji se mogu pronaći u znanstvenim publikacijama.

Svakako najjednostavniji pristup ovom problemu jest izgradnja sustava koji će na temelju ručno podešenih pravila i parametara nastojati izdvojiti koristan tekst od šuma. Mana ovakvog pristupa jest da se za svako web-sjedište moraju razviti posebna pravila kako bi postupak funkcionirao, a i ta pravila je potrebno mijenjati svaki put kad određeno web-sjedište doživi redizajn.

Jedan od prvih i popularnijih algoritama za ekstrakciju sadržaja zove se *Body Text Extraction* ili skraćeno BTE (Finn i dr., 2001). Taj se algoritam zasniva na jednostavnom principu pronalaska kontinuiranog dijela HTML-dokumenta u kojem se nalazi najviše teksta i najmanje HTML-oznaka. Inače je ova mjera vrlo uobičajena u ovoj upotrebi, jer je tipično za koristan tekst da, osim ponešto naredbi za formatiranje, neće sadržavati puno HTML-koda.

Mana algoritma BTE je prvenstveno u inzistiranju na jednom kontinuiranom bloku, a također je problem i kvadratna vremenska složenost algoritma, što ga čini neučinkovitim na većim HTML-dokumentima. Ovi su problemi riješeni u kasnijoj nadogradnji algoritma (Pinto i dr., 2002) naziva DSC (engl. *Document Slope Curves*).

Zanimljiv, drugačiji pristup ovom problemu jest promatranje informacijskog sadržaja, tj. entropije, u segmentima HTML-dokumenta (Lin i Ho, 2002 i Yi i dr., 2003), no uz manje uspjeha. U pravilu su oni dijelovi koji sadrže koristan tekst bogati informacijskim sadržajem, dok se u nebitnim dijelovima često ponavljaju slične strukture.

Za razlikovanje teksta od šuma može poslužiti i omjer hiperveza i teksta u određenom dijelu HTML-dokumenta (konkretno, u dijelu dokumenta sadržanom u određenoj HTML-oznaci). Link Quota Filter (Mantratzis i dr., 2005) je primjer mjere koja za dani segment dokumenta ocjenjuje taj omjer, pa je moguće ekstrakciju teksta temeljiti na traženju onih segmenata kod kojih je taj omjer što manji. Ova mjera je intuitivno jasan indikator onih dijelova stranice koji predstavljaju navigacijske segmente i ne sadrže bitan tekst, no nije dovoljno dobra za cjelokupan

zadatak, pa se koristi uglavnom u drugim algoritmima kao komponenta šireg postupka.

Algoritam *Content Code Blurring* (CCB) zasniva se na pronalaženju homogenih dijelova izvornog koda HTML-dokumenta (Gottron, 2008a).

Postoji i rješenje koje se zasniva na računanju omjera teksta i HTML-oznaka, i to za svaki redak HTML-dokumenta pojedinačno (Weninger i Hsu, 2008 i kasnije unaprijeđeno u Weninger i dr., 2010). Na dobivenim omjerima (engl. *text-to-tag ratio*) primjenjuju se metode nenadziranog grupiranja kako bi se ustanovila razlika između redaka sa sadržajem i redaka sa šumom.

Algoritam VIPS (engl. *a VIsion-based Page Segmentation Algorithm*) (Cai i dr., 2003) specifičan je po tome što se oslanja na karakteristike vizualnog prikaza HTML-dokumenta. No, ova metoda ima dvije značajne mane. Prva je da VIPS sam po sebi ne dijeli otkrivene blokove na zanimljive i nezanimljive, a druga da je za utvrđivanje vizualnih karakteristika neke web-stranice potrebno mnogo više od samog HTML-dokumenta – potrebno je također dohvatiti i datoteke CSS (engl. *Cascading Style Sheets*), koje opisuju izgled pojedinih dijelova teksta, zatim slike i slične popratne sadržaje. To čini VIPS mnogo složenijim algoritmom, kako u implementacijskom smislu, tako i u smislu performansi njegovog izvođenja na velikom skupu web-stranica.

Pobjednik natjecanja CleanEval (Marek i dr., 2007) dijeli HTML-dokument u niz blokova i potom određuje je li neki blok koristan tekst ili šum koristeći metodu uvjetnih slučajnih polja (engl. *conditional random fields*), koju primjenjuje nad nekoliko karakteristika bloka.

Algoritam *Maximum Subsequence Segmentation* (MSS, u prijevodu: podjela po maksimalnim podnizovima) kombinira dva pristupa – heuristički pristup i nadzirano strojno učenje (Pasternack i Roth, 2009). Pošto se temelji na nadziranom strojnom učenju i to na temelju nekih karakteristika teksta unutar web-stranica, MSS posjeduje jednu nepoželjnu karakteristiku, a to je osjetljivost na skup za učenje. Ovisno o naravi, ali i jeziku web-stranica na kojima je istreniran, njegova učinkovitost može varirati od primjene do primjene.

Danas jedan od popularnijih algoritama za ovu primjenu naziva se Boilerpipe (Kohlschütter i dr., 2010). Točnije, radi se o skupini algoritama. U svojem radu autori ocjenjuju tekstne i strukturne karakteristike svakog dijela HTML-dokumenta te na

temelju njih donose odluku. Zasnovani su na stablima odluke i SVM-algoritmu. Bolierpipe ostvaruje vrlo dobre rezultate što ga u kombinaciji s činjenicom da je dostupan kao biblioteka otvorenog izvornog koda (engl. *open source*) čini vrlo popularnim.

U svojem doktorskom radu Pomikalek (2011) prezentira nov algoritam, koji naziva jusText, i koji pokazuje dobre rezultate. Zasniva se na izdvajanju iz HTML-dokumenta onih segmenata koji su sadržani u HTML-elementima koji tipično definiraju blokove teksta. U tako izdvojenim blokovima onda se promatraju karakteristike poput omjera poveznica u odnosu na ostatak teksta, ali promatraju se i karakteristike teksta te se po upotrebi određenih riječi nastoji utvrditi radi li se tu o gramatički oblikovanom tekstu, u kojem slučaju se smatra bitnim. Algoritam uzima u obzir i susjedne blokove, krenuvši od pretpostavke da blokovi zanimljivog teksta, kao i blokovi nezanimljivog, dolaze u skupinama.

Za razliku od prethodno opisanih rješenja koja ekstrakciju temelje na informacijama sadržanima u samom HTML-dokumentu koji obrađuju, postoje i rješenja koja pokušavaju iskoristiti činjenicu da unutar istog web-sjedišta velik broj stranica obično ima vrlo sličnu strukturu (Bar-Yossef i Rajagopalan, 2002). Taj pristup počiva na pronalaženju onih dijelova HTML-dokumenta koji se ponavljaju u više stranica unutar istog web-sjedišta, tj. na pronalaženju predloška iz kojeg je stvorena stranica. Upotreba predložaka za automatizirano stvaranje web-stranica na temelju dijelova teksta dobivenih iz baze podataka na internetu je vrlo rasprostranjena. Mana takvog pristupa jest u tome što je prije same ekstrakcije potrebno pronaći nekoliko primjeraka web-stranica istog web-sjedišta koje su izrađene iz istog predloška kako bi algoritam ustanovio koje dijelove valja odbaciti. To nije nužno uvijek moguće niti lako.

Postoje i sustavi za ekstrakciju sadržaja koji omogućuju kombiniranje više drugih algoritama, primjerice sustav CombinE (Gottron, 2008b). Korisnik sustava bira, po želji, koje će algoritme koristiti i na koji način će ih organizirati – mogu se koristiti serijski, gdje rezultat jednog postaje ulaz drugog, ili paralelno, gdje se rezultati dvaju algoritama na određeni način kombiniraju da bi se dobio konačan rezultat.

Ima radova u ovom području koji se bave web-stranicama na hrvatskom jeziku. Ljubešić i Erjavec (2011) u svom radu pristupaju izgradnji tekstnog korpusa za hrvatski i slovenski jezik. Osim samog korpusa, za čiju su izgradnju prikupili gotovo cjelokupni web na hrvatskom i slovenskom jeziku, razvili su i vlastiti algoritam ekstrakcije sadržaja kojeg u svom radu uspoređuju s algoritmima BTE i Boilerpipe. Ukoliko gledamo ocjenu F1, Boilerpipe se u njihovom ispitivanju pokazao najboljim, s rezultatom od 0.882, no njihov algoritam, zasnovan na traženju što većeg komada teksta koji se nalazi na istoj dubini u DOM-stablu (engl. *Document Object Model*, hijerarhijski model HTML-dokumenta gdje HTML-oznake predstavljaju čvorove stabla), ostvaruje vrlo blizak rezultat od 0.821, no uz iznimno visoku preciznost (0.979) i stoga nešto slabiji odaziv (0.707). Boilerpipe je pak ujednačen u te dvije mjere, uz nešto viši odaziv (0.921) od preciznosti (0.847). BTE ostvaruje najviši odaziv (0.955), ali mu je ocjena F1 najniža, samo 0.713, zbog vrlo niske preciznosti (0.570).

Krišto (2009) se u svojem radu koncentrira na problem ekstrakcije sadržaja. Razvio je dva algoritma, jedan temeljen na prepoznavanju nebitnog sadržaja pomoću određenih identifikacijskih atributa HTML-oznaka te drugi, koji analizira odnose gustoće HTML-elemenata i teksta u dokumentu te pokušava izdvojiti segment u kojem prevladava čisti tekst. U oba slučaja dobiva sličan rezultat: ocjena F1 oba algoritma je u ispitivanju bila 0.76, te oba algoritma daju dobar odaziv, ali nisku preciznost.

4. Opis postupka

U ovom se radu koriste neki postojeći algoritmi za ekstrakciju tekstnog sadržaja i to posebno oni za koje postoji implementacija za Microsoftovu programsku platformu .NET Framework, uz iznimku metode Body Text Extraction za koju je izvedena vlastita implementacija prema opisu algoritma u izvornom članku (Finn i dr., 2001).

Nastojalo se izvesti kombinaciju tih osnovnih ekstraktora uz pomoć metoda nadziranog strojnog učenja s ciljem postizanja boljeg rezultata od korištenja osnovnih algoritama zasebno. Izvedene su dvije varijante postupka kombinacije. U prvoj se nastoji ostvariti klasifikator koji će na temelju određenih značajki cijelog HTML-dokumenta nadzirano naučiti koji je od pojedinačnih ekstraktora najprikladniji za dani dokument. Rezultat prve varijante uspoređuje se s rezultatom samih osnovnih algoritama ekstrakcije. U drugoj varijanti HTML-dokument se rastavlja na segmente i nastoji nadzirano naučiti koji su segmenti zanimljivi, koristeći pritom značajke segmenata i dodatnu informaciju o tome smatraju li osnovni ekstraktori taj segment zanimljivim. Rezultat druge varijante uspoređuje se također s rezultatom osnovnih ekstraktora i dodatno s ekstraktorom, nazvanim segmentnim ekstraktorom, koji je nadzirano naučen klasificirati segmente samo po njihovim značajkama, bez informacije o odlukama osnovnih ekstraktora.

Također je razvijena i metoda ekstrakcije tekstnog sadržaja na temelju jednostavnih predložaka. Predlošci se zadaju u tekstnoj datoteci i opisuju elemente HTML-dokumenta čiji će tekstni sadržaj biti ekstrahiran.

U sljedećim potpoglavljima prvo će biti opisani osnovni, postojeći algoritmi ekstrakcije koji su upotrijebljeni u ovom radu. Zatim slijedi opis vektora značajki HTML-dokumenata i HTML-segmenata koji se koristi u hibridnim ekstraktorima, te potom detaljniji opis načina na koji su kombinirani osnovni ekstraktori u dvjema varijantama hibridnog ekstraktora sadržaja. Bit će opisan i segmentni ekstraktor, te postupak vrednovanja i skup podataka koji je korišten u ispitivanju. Na kraju slijedi opis izvedene metode ekstrakcije po predlošku.

4.1. Korišteni algoritmi za ekstrakciju teksta iz web-stranica

U većini slučajeva radi se o implementacijama algoritama opisanih u poglavlju 3, a koje su javno dostupne na webu. Ući će se u nešto više detalja o načinu rada njihovih algoritama.

Pored ovih biblioteka postoje i mnogi komercijalni i nekomercijalni samostalni gotovi proizvodi te mnoga rješenja koja su temeljena na webu i nude se kao usluge kojima se pristupa kroz Web API (engl. *Application Programming Interface*), no takva je rješenja teže koristiti u obradi velikog broja web-stranica pošto je HTML-dokumente potrebno slati na obradu preko weba do poslužitelja te čekati na rezultat, što cijeli postupak čini značajno sporijim.

4.1.1. Boilerpipe

Boilerpipe je veoma popularna i vrlo učinkovita biblioteka otvorenog izvornog koda (engl. *open source library*) koja implementira istoimenu skupinu algoritama. Neke od karakteristika tekstnih blokova koje Boilerpipe izračunava i na temelju kojih donosi odluku su:

- prisutnost fraza koje se redovito ponavljaju u nezanimljivim dijelovima teksta;
- prisutnost određenih HTML-oznaka kojima se obično omeđuju blokovi korisnog teksta: <hX> naslov, <p> paragraf, <a> poveznica, odjeljak;
- "plitke" karakteristike teksta: prosječna duljina riječi i rečenica, broj riječi u cijelom segmentu;
- lokalni kontekst: apsolutni i relativni položaj teksta u stranici;
- heuristička pravila: broj riječi koje počinju velikim slovom, broj riječi napisanih sa svim velikim slovima, broj oznaka datuma i vremena u tekstu, gustoća linkova, te neki omjeri prethodnih veličina;
- gustoća teksta u segmentu.

Boilerpipe koristi SVM (engl. *Support Vector Machines*) i stabla odluke, koji su naučeni na skupu ručno obrađenih primjera za učenje.

U ovom radu upotrijebljena su dva algoritma iz skupine algoritama Boilerpipe – Boilerpipe Default Extractor i Boilerpipe Article Extractor. Prvi je namijenjen za zadaću ekstrakcije teksta iz općenitog HTML-dokumenta bilo kakve strukture, dok je

potonji posebno prilagođen za ekstrakciju teksta iz novinskih i drugih članaka objavljenih na webu.

4.1.2. Readability

Readability, originalno razvijen u programskom jeziku JavaScript kao dodatak za web-preglednik, danas je dostupan kao biblioteka u verzijama za razne platforme. Također je u vrlo širokoj upotrebi. Originalna namjena mu je bila olakšati čitanje članaka na stranicama na kojima previše reklama i nebitnog sadržaja ometa ugodno čitanje.

Zasniva se na posebno prilagođenim, ručno razvijenim heurističkim pravilima. Može prepoznati i izdvojiti naslov, tekst članka i prateće ilustracije.

U primjeni u ovom radu rezultat biblioteke Readability je nakon njenog završetka obrade dodatno očišćen od HTML-oznaka, pošto Readability u svojoj originalnoj verziji ne proizvodi čisti tekst kao rezultat, već novi HTML-dokument namijenjen prikazu u web-pregledniku.

4.1.3. Body Text Extraction

Body Text Extraction (BTE) je odabran jer je, kao jedan od prvih algoritama za ovu namjenu, često korišten, a i u usporedbi s kasnije razvijenim algoritmima još uvijek ostvaruje dobre rezultate. Također mu je prednost relativna jednostavnost pristupa problemu, što doprinosi njegovoj otpornosti na promjene u specifikaciji HTML-jezika i načinima izrade HTML-dokumenata na webu, koji su se oboje dosta promijenili od vremena nastanka algoritma.

Kako za BTE ne postoji implementacija za platformu .NET, za potrebe ovog rada je izvedena implementacija u programskom jeziku C#.

BTE je zasnovan na principu pronalaženja najvećeg kontinuiranog segmenta HTML-dokumenta unutar kojega je što manje HTML-oznaka, a izvan njega što više (Finn i dr., 2001). BTE u prvom koraku rastavlja HTML-dokument na niz tokena, gdje svaki pojedini token može biti riječ teksta ili HTML-oznaka. Svakom se tokenu dodijeljuje brojčana vrijednost i to tako da se oznakama dodijeljuje vrijednost 1, a riječima 0. Ovako dobiveni niz vrijednosti označen je sa B_i a njegova duljina sa N. Nad tim nizom traže se takve vrijednosti i i j za koje će sljedeća vrijednost biti maksimizirana:

$$T_{i,j} = \sum_{n=0}^{i-1} B_n + \sum_{n=i}^{j} (1 - B_n) + \sum_{n=j+1}^{N-1} B_n$$

Tako dobiveni *i* i *j* predstavljaju početnu i završnu granicu segmenta HTML-dokumenta iz kojeg je još samo potrebno ukloniti sve oznake koje se nalaze unutar njega.

4.2. Izvedba hibridnih ekstraktora

Svaka od prethodno navedenih metoda ima određene prednosti i mane o kojima ovisi uspješnost njezine ekstrakcije teksta iz nekog HTML-dokumenta. Ideja je pokušati, upotrebom algoritama strojnog učenja, stvoriti takav ekstraktor sadržaja koji bi se oslanjao na postojeće metode ekstrakcije, ali bi uz pomoć istreniranog klasifikatora na određeni način birao, ovisno o karakteristikama HTML-dokumenta ili nekog njegovog segmenta, koju metodu primijeniti u kojem slučaju s ciljem ostvarivanja boljih rezultata od svake od metoda zasebno.

Za potrebe ostvarivanja ovih klasifikatora odabran je algoritam SVM (*Support Vector Machine*, Cortes i Vapnik, 1995).

4.2.1. Vektori značajki HTML-dokumenta i segmenta

Oba klasifikatora na ulazu primaju određene kvantificirane značajke HTML-dokumenta, odn. segmenta dokumenta. Svojstva koja se promatraju odabrana su na način da što bolje predstavljaju strukturne karakteristike HTML-dokumenta koje bi mogle biti relevantne u donošenju odluka.

Vektor značajki HTML-dokumenta/segmenta sadrži sljedeće:

- Veličina HTML-dokumenta/segmenta, kao multinomijalna značajka koja je u izlaznom vektoru prikazana sa šest binarnih značajki na način da se na jednom od tih šest mjesta nalazi jedinica, a na ostalima nula, ovisno o tome koji je red veličine HTML-dokumenta:
  - prvi broj je 1 ako je veličina manja od 100 bajtova;
  - drugi broj je 1 ako je veličina veća ili jednaka 100 i manja od 1.000 bajtova;

- treći broj je 1 ako je veličina veća ili jednaka 1.000 i manja od 10.000 bajtova;
- četvrti broj je 1 ako je veličina veća ili jednaka 10.000 i manja od 100.000 bajtova;
- peti broj je 1 ako je veličina veća ili jednaka 100.000 i manja od 1.000.000 bajtova;
- šesti broj je 1 ako je veličina veća ili jednaka 1.000.000 bajtova;

- Omjer ukupne duljine HTML-oznaka i duljine cijelog HTML-dokumenta/segmenta;
- Omjer ukupne duljine oznaka poveznica i duljine cijelog HTML-dokumenta/segmenta;
- Tri mjere distribucije blokova teksta u dokumentu dobivene na način da se iz HTML-dokumenta/segmenta izdvoje svi kontinuirani blokovi u kojima se nalazi samo tekstni sadržaj, bez HTML-oznaka. Nisu uzeti u obzir dijelovi teksta koji se nalaze unutar poveznica pošto oni često ne predstavljaju zanimljiv sadržaj – većina se nalazi u navigacijskim izbornicima i sl. – i uglavnom su vrlo kratki, pa bi s obzirom na svoju brojnost značajno utjecali na ove mjere. Potom iz tako pripremljenih blokova računamo tri mjere:
  - prosječna duljina blokova teksta podijeljena s duljinom najduljeg bloka;
  - medijan duljina blokova teksta podijeljen s duljinom najduljeg bloka;
  - duljina najduljeg bloka teksta podijeljena sa zbrojem duljina svih blokova teksta;
- Frekvencije HTML-oznaka – utvrđen je broj pojavljivanja određenih oznaka koje imaju utjecaja na strukturu HTML-dokumenta/segmenta, i tako dobiveni niz brojeva normaliziran je na način da je svaki član podijeljen s iznosom najvećeg od njih, čime je svaka vrijednost ograničena na raspon od 0 do 1.
  - Promatrane su sljedeće HTML-oznake: blockquote, br, caption, cite, code, col, colgroup, div, dl, form, frameset, h1, h2, h3, h4, h5, h6, hr, iframe, object, ol, p, samp, span, table i ul.

Na ovaj se način dobivaju vektori od 37 značajki. Svaka od pojedinih brojki u vektoru kreće se u rasponu od 0 do 1, što je bitno za primjenu u algoritmu SVM

kako ne bi neke od značajki imale disproporcionalno velik utjecaj na rezultat klasifikacije.

4.2.2. Prva varijanta hibridnog ekstraktora

Prva varijanta hibridnog klasifikatora nastoji postići mogućnost predviđanja toga koja će od raspoloživih metoda postići najbolji rezultat na određenom HTML-dokumentu. Sa tako naučenim klasifikatorom HTML-dokumenata bi se, idealno, na većem skupu HTML-dokumenata postizao bolji učinak u ekstrakciji od učinka koji se postiže korištenjem jedne od metoda zasebno na cijelom skupu HTML-dokumenata.

Za učenje klasifikatora potrebno je prvo na skupu HTML-dokumenata odvojenih za učenje primijeniti svaku od pojedinačnih metoda ekstrakcije koje ovdje koristimo te vrednovati njihove rezultate u odnosu na zlatni standard. Potom se za svaki HTML-dokument iz tog skupa računa njegov vektor značajki. Ti se vektori daju SVM-u kao ulaz u procesu nadziranog učenja. Ciljana kategorija u koju se želi da SVM klasificira neki dokument određena je time koja je od metoda za taj dokument imala najbolju ocjenu F1. Konkretno, ako je BTE metoda ostvarila najbolju ocjenu F1, oznaka kategorije će biti 0, ako je Readability metoda bila najbolja, oznaka će biti 1, ako je Boilerpipe Default Extractor bio najbolji, oznaka će biti 2, i ako je Boilerpipe Article Extractor bio najbolji, oznaka će biti 3. Na ovaj način dobiva se naučeni model koji se kasnije koristi u ispitivanju.

U fazi ispitivanja za svaki se HTML-dokument iz skupa za ispitivanje računa vektor značajki. Taj se vektor daje prethodno naučenom modelu koji na temelju tih značajki daje predviđanje koji će algoritam biti najučinkovitiji na tom dokumentu. Potom se primjenjuje taj algoritam i pohranjuje rezultat.

4.2.3. Druga varijanta hibridnog ekstraktora

U drugoj varijanti su HTML-dokumenti rastavljani na segmente te su u vektor značajki, pored prethodno opisanih, dodane još četiri značajke koje opisuju u kolikoj je mjeri tekst koji se nalazi unutar danog segmenta zastupljen u izlazu ekstrakcije određene osnovne metode. Potom se pokušalo istrenirati takav klasifikator koji bi dakle na temelju karakteristika jednog segmenta HTML-dokumenta i sa dodatnom informacijom o tome smatraju li četiri osnovna klasifikatora taj segment zanimljivim donosio konačnu odluku o tome je li dotični segment zanimljiv ili ne i na taj način konstruirao izlazni, čisti tekst.

Rastavljanje HTML-dokumenata na segmente nije trivijalan zadatak i za to je bilo potrebno razviti poseban algoritam. Algoritam prolazi kroz dijelove HTML-dokumenta, posebno izdvajajući "div" i "table" elemente, koji se obično koriste za strukturiranje sadržaja, u odvojene segmente. Drugi elementi i tekstni blokovi grupiraju se zajedno u segment, no ukoliko se ustanovi da neki HTML-element, bez obzira na vrstu, sadrži više od pola teksta koji je ukupno sadržan u dokumentu, algoritam se rekurzivno primjenjuje na njegov sadržaj. Više detalja o algoritmu razdvajanja bit će dano u poglavlju s opisom implementacije.

Dodatne četiri značajke segmenta opisuju u kolikoj je mjeri tekst unutar segmenta zastupljen u izlazu pojedine osnovne metode ekstrakcije. Zastupljenost se mjeri tako da se utvrdi duljina najduljeg zajedničkog podniza između teksta u segmentu i teksta koji je ekstrahirala pojedina metoda, te tako dobiven broj riječi podijeli s brojem riječi u samom segmentu da bi se dobio broj u rasponu od 0 do 1. Ukoliko je pak broj riječi teksta u segmentu jednak nuli, smatra se da je poklapanje jednako 0.

U fazi učenja potrebno je naučiti SVM da prepoznaje zanimljive segmente teksta. Klasifikator dakle ima zadaću razvrstati segmente u dvije kategorije: zanimljiv tekst i nezanimljiv tekst. Za potrebe određivanja hoće li se segment smatrati zanimljivim, promatra se na isti način kao gore zastupljenost teksta u segmentu u tekstu zlatnog standarda. Ako je mjera zastupljenosti veća od 0.9, smatra se da je segment zanimljiv. Do iznosa ovog praga, koji se može nazvati pragom ekstrakcije, došlo se eksperimentiranjem s raznim vrijednostima u rasponu od 0.5 do 0.9. Pokazalo se da su rezultati u tom rasponu vrlo slični, pa je odabrana višu vrijednost kako bi se na taj način kompenzirala nepreciznost izabrane metode mjerenja zastupljenosti.

U fazi ispitivanja se HTML-dokumenti rastavljaju na segmente te se za svaki segment na isti način gradi vektor značajki. Ukoliko ga istrenirani SVM svrsta u kategoriju zanimljivih segmenata, preuzima se tekst iz tog segmenta. Na taj način, segment po segment, sastavlja se rezultat ekstrakcije.

4.3. Segmentni ekstraktor

Pošto druga varijanta hibridnog klasifikatora puno manje ovisi o rezultatima drugih metoda – njihovi rezultati proizvode samo četiri od 41 značajke na ulazu u SVM klasifikator – bilo je moguće uz minimalne izmjene od nje napraviti takav ekstraktor koji bi odluku o tome da li će smatrati neki HTML-segment zanimljivim ili

ne donositi samo na temelju značajki samog HTML-segmenta, bez dodatnih informacija o tome kako taj segment ocjenjuju četiri osnovne metode.

Izvedeno je ispitivanje i s ovakvim ekstraktorom kako bi se usporedio rezultat s onim koji se dobiva iz druge varijante hibridnog ekstraktora. Na taj način može se ocijeniti kakav je utjecaj dodatna informacija o rezultatu osnovnih ekstraktora imala na drugu varijantu hibridnog ekstraktora.

4.4. Postupak vrednovanja

O samim ocjenama koje se koriste za mjerenje uspješnosti ekstrakcije govorilo se u drugom poglavlju. Ovdje će biti opisan konkretan postupak koji se koristi u ovom radu.

Za potrebe uspoređivanja ekstrahiranog teksta i zlatnog standarda u prvom koraku koristi se postupak opisan u drugom poglavlju gdje se oba teksta rastavljaju u niz riječi. Prije samog rastavljanja iz oba se teksta uklanjaju možebitne preostale HTML-oznake, a potom se vrši razdvajanje na način da znakovi praznine omeđuju riječi.

U postupku pripreme preskaču se svi znakovi u tekstu čija je kodna vrijednost veća od 127. Na ovaj se način ulaz ograničava samo na znakove iz skupa znakova ASCII, i izbjegava mogućnost da razlika u načinu kodiranja teksta pojedinih algoritama utječe na njihove ocjene. Iako su ovime iz teksta izbačeni i hrvatski dijakritički znakovi, pošto se nijedna riječ ne sastoji samo od takvih znakova te pošto je osim samih slova važan i položaj riječi u odnosu na ostale riječi u tekstu, ova promjena ne može imati primjetnog utjecaja na rezultat evaluacije.

Da bi se pronašla duljina najvećeg zajedničkog podniza ekstrahiranog teksta i zlatnog standarda, upotrijebit će se općeniti algoritam za ponalaženje duljine najduljeg zajedničkog podniza koji je opisan u (Hirschberg, 1975). To je klasični algoritam za rješavanje ovog problema, iz obitelji algoritama dinamičkog programiranja, koji je u Hirschbergovoj verziji dodatno poboljšan na način da mu je smanjena prostorna složenost izvođenja.

Nakon izračunavanja ocjena za svaki pojedini dokument, na način opisan u drugom poglavlju, računa se i ocjena nad cijelim skupom, kao odgovarajuće postavljena težinska prosječna vrijednost svih pojedinačnih rezultata.

Iz ukupnih ocjena preciznosti i odziva, dobivenih kao težinski prosjek pojedinačnih ocjena, računa se ukupna F1-ocjena algoritma nad cijelim skupom. Tako dobivena vrijednost koristi se kao ukupna ocjena uspješnosti metode.

4.5. Ispitni skupovi podataka

Za ispitivanje algoritama za ekstrakciju tekstnog sadržaja, te za učenje i ispitivanje hibridnog algoritma, korišten je skup podataka stvoren za potrebe natjecanja CleanEval (Baroni i dr., 2008). Taj skup na žalost više nije dostupan na izvornom web-sjedištu, pa je korištena njegova verzija koja je pronađena na web-stranicama[*] rada (Weninger i dr., 2010) o kojem se pisalo u poglavlju o metodama ekstrakcije tekstnog sadržaja.

Skup podataka sadrži 935 HTML-dokumenata i 736 datoteka sa zlatnim standardima. U njemu su dodatno sadržani i dokumenti na kineskom jeziku, no oni nisu uključeni u ispitivanje. Za neke dokumente u skupu nije bio prisutan zlatni standard, a kako je uvidom u sadržaj tih HTML-dokumenata bilo očigledno da sasvim sigurno sadrže zanimljiv tekstni sadržaj, ti su dokumenti uklonjeni iz ispitnog skupa da ne bi doprinosili pogrešnom snižavanju ocjena uspješnosti algoritama ekstrakcije. Konkretno, svaki dokument iz kojeg metoda uspije izvući određeni sadržaj, a taj dokument nema zlatni standard, metodi smanjuje mjeru preciznosti.

Svaki HTML-dokument u skupu CleanEval sadržavao je jednu dodatnu pseudo-HTML-oznaku, kakva nije podržana HTML-standardom već je dodana samo za potrebe čuvanja određenih metapodataka o HTML-dokumentu. Ta je oznaka, naziva "text", uklonjena iz HTML-dokumenata. U dokumentima zlatnog standarda nalazili su se također neki metapodaci, primjerice URL HTML-dokumenta iz kojeg je zlatni standard izvučen, što je također uklonjeno. U ostatku teksta zlatnog standarda nalazile su se rudimentarne oznake kojima su označeni neki metapodaci o naravi dijelova izvučenog teksta (primjerice, radi li se o naslovu, elementu neke liste, ili o odjeljku teksta), no oni su zadržani pošto je korištena metoda evaluacije neosjetljiva na preostale oznake u ekstrahiranom tekstu i tekstu zlatnog standarda.

Nakon ove obrade dobivena su 723 HTML-dokumenta u parovima sa 723 dokumenta zlatnog standarda. Iz tog je skupa potom izdvojeno 300 parova dokumenata u skup za treniranje za potrebe prve varijante. Za drugu je varijantu izdvojeno samo 60 dokumenata iz cijelog skupa, pošto se nakon rastavljanja na

[*]Dostupno na web-adresi: http://web.engr.illinois.edu/~weninge1/cetr/

segmente dobije mnogo više ulaznih vektora značajki, pa je postupak učenja na skupu od 300 dokumenata bio prespor.

4.6. Ekstrakcija na temelju predloška

U sklopu ovog rada razvijena je i metoda ekstrakcije koja svoj rad temelji na korisnički definiranom predlošku. Cilj je bio postići lakše korištenje, pa je stoga podržani oblik definiranja predloška dosta jednostavan.

Predložak se definira u tekstnoj datoteci. Svaki redak te datoteke predstavlja jedan selekcijski izraz. Ekstraktor će na temelju tih selekcijskih izraza pronaći sve HTML-elemente koji zadovoljavaju barem jedan od uvjeta definiranih selekcijskim izrazima.

Podržane su tri vrste selekcijskih izraza – selekcija elementa, selekcija elementa s atributom i selekcija elementa s atributom koji ima zadanu vrijednost.

Selekcija elementa je najjednostavniji selekcijski izraz. Definira se retkom u datoteci predloška koji sadrži samo ime HTML-elementa koji treba selektirati. Sav tekstni sadržaj elemenata tog imena će biti ekstrahiran.

Selekcija elementa s atributom definira se retkom u datoteci predloška koji sadrži ime nekog HTML-atributa nakon kojega se nalazi znak jednakosti. Sav tekstni sadržaj elemenata koji imaju atribut tog imena, bez obzira na konkretnu vrijednost tog atributa na pojedinom elementu, bit će ekstrahiran. Primjerice, izraz "href=" će selektirati sve HTML-elemente koji imaju definiran atribut "href".

Selekcija elementa s atributom koji ima zadanu vrijednost definira se retkom u datoteci predloška koji ima oblik "ime=vrijednost". Na temelju takvog selekcijskog izraza ekstraktor će pronaći sve HTML-elemente koji imaju atribut imena "ime" i kojem je dodijeljena vrijednost "vrijednost", te ekstrahirati sav njihov tekstni sadržaj. U selekcijskom izrazu vrijednost se navodi bez ikakvih dodatnih graničnika, dakle, nije ju potrebno stavljati u navodnike ili sl. Sve što se nalazi iza znaka jednakosti očekuje se pronaći u vrijednosti atributa. Primjerice, izraz "class=content" će selektirati sve elemente koji imaju atribut "class" i kojem je postavljena vrijednost "content".

Ukoliko se neki od selektiranih elemenata nalazi unutar nekog drugog također selektiranog elementa, takav pod-element se preskače, pošto je njegov tekstni sadržaj već obuhvaćen prilikom ekstrakcije tekstnog sadržaja nad-elementa.

5. Rezultati

Ispitana je učinkovitost razvijenih varijanti hibridnih ekstraktora na već opisanom skupu podataka koji potječe iz natjecanja CleanEval. Također je nad skupovima za ispitivanje izvršena ekstrakcija pomoću četiri zasebna algoritma koji su u ovom radu bili kombinirani i ekstrakcija pomoću segmentnog ekstraktora.

5.1. Prva varijanta hibridnog ekstraktora

Za prvu je varijantu pripremljen skup od 300 HTML-dokumenata za učenje i 423 dokumenta za ispitivanje. Njena je zadaća bila naučiti možebitnu međuovisnost između strukture dokumenta, izražene kroz prethodno opisanih 37 značajki, i metode koja će nad takvim dokumentom ostvariti najbolji rezultat. Stoga je u njenom učenju bilo potrebno prvo izvršiti ekstrakciju cijelog skupa za učenje s osnovnim metodama te evaluirati njihov učinak na tom skupu. U Tablici 1 dan je rezultat pojedinačnih algoritama nad cijelim skupom za učenje.

Tablica 1: Rezultat evaluacije osnovnih metoda nad skupom za učenje prve varijante hibridnog ekstraktora

Metoda	Preciznost	Odaziv	F1
BTE	0.9329	0.9655	0.9489
Boilerpipe AE	0.971	0.5636	0.7132
Boilerpipe DE	0.9594	0.6414	0.7688
Readability	0.9530	0.8008	0.8703

Iznenađujuće, algoritam Body Text Extraction ostvario je najbolji F1-rezultat nad cijelim skupom, s ujednačenim mjerama preciznosti i odaziva. Tu možemo naslutiti razlog njegove dugotrajne popularnosti u području ekstrakcije tekstnog sadržaja. Oba Boilerpipe algoritma imaju visoku preciznost, višu od algoritma BTE, no njihov je odaziv slab, što je smanjilo konačnu F1-ocjenu. Slično je i s algoritmom Readability, koji ipak zahvaljujući nešto boljem odazivu ostvaruje drugi najbolji F1-rezultat.

S dobivenim detaljnim rezultatima vrednovanja određuje se najuspješnija metoda, po F1-ocjeni, za svaki HTML-dokument. To predstavlja željeni izlaz SVM klasifikatora. Nakon treniranja dobiva se SVM model koji se primjenjuje na skup podataka za ispitivanje. Također je nad skupom za ispitivanje izvršena ekstrakcija pomoću svake pojedinačne metode radi usporedbe. Tablica 2 prikazuje rezultate svih primijenjenih ekstraktora.

Tablica 2: Rezultat ekstrakcije osnovnih metoda i prve varijante hibridnog ekstraktora nad skupom za ispitivanje

Metoda	Preciznost	Odaziv	F1
BTE	0.9265	0.9614	0.9437
Boilerpipe AE	0.9597	0.5546	0.703
Boilerpipe DE	0.9525	0.6322	0.76
Readability	0.9465	0.8117	0.8739
SVM-A	0.9253	0.9348	0.9301

Vidi se da je rezultat osnovnih metoda, očekivano, vrlo sličan rezultatu koji je dobiven nad skupom za učenje. Prva varijanta hibridnog ekstraktora, SVM-A, ostvaruje dosta dobar rezultat, sa drugom najboljom F1-ocjenom, no u svim je mjerama neznatno lošija od, ponovno najboljeg, algoritma BTE.

Na prvi pogled ovo se čini kao dobar rezultat, no dubljom analizom predviđanja koja je dao SVM-klasifikator, uočeno je kako je u gotovo svim slučajevima, uz samo 7 iznimaka, klasifikator odabrao algoritam BTE za metodu koja će biti primijenjena na HTML-dokumentima skupa za ispitivanje. To objašnjava veliku sličnost njegovog rezultata s rezultatom algoritma BTE. Analiza rezultata evaluacije svih algoritama na pojedinim dokumentima ipak otkriva kako bi, da je kojim slučajem SVM-klasifikator činio idealan izbor metode ekstrakcije, njegova predviđanja bila značajno manje monotona.

Iako je teško razlučiti kakvu je točno pravilnost SVM-model izvukao iz podataka u skupu za učenje, ovakvo ponašanje klasifikatora nad ispitnim skupom upućuje na zaključak kako klasifikator, u osnovi, nije uspio izvući nikakvu korisnu pravilnost, osim te da je BTE u prosjeku najbolji izbor, pa stoga u gotovo svim slučajevima bira tu metodu. Željeni cilj ostvarivanja boljeg rezultata od primjene pojedinačnih metoda

ovom varijantom dakle nije ostvaren – nije uspješno stvoren takav klasifikator koji bi na temelju odabranih značajki HTML-dokumenta predviđao uspješnost pojedinih metoda ekstrakcije.

5.2. Druga varijanta hibridnog ekstraktora i segmentni ekstraktor

Za ispitivanje druge varijante hibridnog ekstraktora iz skupa CleanEval izdvojeno je samo 60 HTML-dokumenata u skup za učenje, a ostalih 663 su ostavljeni u skupu za ispitivanje. Razlog odabira manjeg broja dokumenata za učenje ove varijante je u činjenici da ova varijanta svoj rad zasniva na segmentima HTML-dokumenata. Utvrđeno je da iz jednog HTML-dokumenta nastaje u prosjeku oko 16 segmenata, što ulazni skup podataka za učenje SVM-klasifikatora čini jako velikim i postupak njegovog učenja presporim na prevelikom skupu. Iz nasumično odabranih 60 dokumenata je, primjerice, odvajanjem u segmente dobiveno ukupno 1055 segmenata, što je sasvim dovoljno za učenje SVM-klasifikatora.

Ovdje rezultati samih osnovnih algoritama nad skupom za učenje nisu posebno zanimljivi zbog osobine druge varijante hibridnog klasifikatora da svoj rad temelji na segmentima, a ne cijelim dokumentima, pa uspješnost pojedinih metoda nad cijelim dokumentima ne mora imati direktnu, jednostavnu vezu s uspjehom tih metoda nad segmentima, pa dakle niti direktan utjecaj na učenje klasifikatora.

U Tablici 3 prikazani su rezultati koje ostvaruju pojedini osnovni algoritmi, druga varijanta hibridnog ekstraktora i segmentni ekstraktor, nad skupom za ispitivanje.

Tablica 3: Rezultat ekstrakcije osnovnih metoda i druge varijante hibridnog ekstraktora nad skupom za ispitivanje

Metoda	Preciznost	Odaziv	F1
BTE	0.9272	0.9595	0.9431
Boilerpipe AE	0.9637	0.5538	0.7034
Boilerpipe DE	0.9551	0.6329	0.7613
Readability	0.9486	0.8056	0.8713
SVM-B	0.886	0.9469	0.9154
SVM-seg	0.9478	0.8583	0.9008

Vidi se da je rezultat osnovnih metoda opet sličan. Druga varijanta hibridnog ekstraktora, SVM-B, ostvarila je dobar rezultat. Iako ima nešto slabiju preciznost od svih drugih metoda, odaziv je visok, tek nešto niži od najboljeg koji ostvaruje algoritam BTE, i stoga ovaj hibrid ostvaruje drugu najbolju F1-ocjenu.

Segmentni ekstraktor pokazuje pak također vrlo dobar F1-rezultat, tek nešto malo slabiji od SVM-B F1 rezultata. Vidimo da je ekstraktor SVM-B, s klasifikatorom koji je primao informacije i o rezultatu osnovnih ekstraktora nad segmentima, ostvario viši odaziv od segmentnog klasifikatora koji nije primao te informacije. No, isti taj SVM-B ekstraktor ostvario je i otprilike jednako nižu preciznost.

Uvidom u točne duljine ekstrahiranog teksta SVM-B i segmentnog ekstraktora uočeno je kako je segmentni ekstraktor izvukao dosta manje teksta, manje i od ukupne veličine zlatnog standarda, dok je SVM-B izvukao više od te veličine. Time se može donekle objasniti slabiji odaziv segmentnog ekstraktora i slabija preciznost druge varijante hibridnog ekstraktora.

Očito je da odabrane strukturne značajke HTML-segmenta već same po sebi daju vrlo korisne informacije u utvrđivanju je li neki segment zanimljiv ili nije. No, ovaj rezultat također pokazuje kako informacija o rezultatima drugih metoda može bitno utjecati na rezultate ekstrakcije te doprinijeti i postizanju boljeg rezultata.

U konačnici nijedna od tri ispitane metode nije uspjela ostvariti rezultat koji bi nadmašio sve pojedinačne metode, tj. algoritam BTE koji je bio najbolji. Dakle, hibridno rješenje se ne isplati.

6. Opis programske izvedbe

Programsko rješenje za potrebe ovog rada izvedeno je u Microsoftovoj .NET Framework tehnologiji, u programskom jeziku C#. Odlučeno je koristiti ovu programsku platformu zbog raspoloživosti velikog broja kvalitetnih biblioteka za razne zadaće. Program uključuje grafičko sučelje, izvedeno u biblioteci Windows.Forms radnog okružja .NET Framework, koje omogućuje lakši i ugodniji rad s programom.

Za potrebe analiziranja HTML-dokumenata korištena je biblioteka otvorenog koda imena Html Agility Pack. Ova biblioteka omogućuje brz i pouzdan rad s HTML-dokumentima i otporna je na mnoge uobičajene greške koje su prisutne u dokumentima na webu.

Za potrebe klasifikacije korištena je biblioteka otvorenog koda koja implementira SVM klasifikator koja je prezentirana u radu (Chang i Lin, 2011). Ova je biblioteka izvorno razvijena u programskim jezicima C++ i Java, no dostupna je i u verziji za .NET Framework[*].

Jedno od načela koja se nastojalo slijediti u izradi programa jest da rezultat svake operacije, u svakom koraku, bude moguće lako proučavati odvojeno od samog programa, u drugim alatima. Stoga je izbjegnuta upotreba ikakvih novih, vlastitih formata pohrane podataka, umjesto toga se oslanjajući na postojeće, jednostavne i široko prihvaćene standarde.

Dijelovi implementacije su, zbog svoje složenosti, izvedeni koristeći Task.Parallel biblioteku .NET Frameworka. Ta biblioteka omogućuje izvršavanje više zadataka paralelno čime se postiže bolje iskorištavanje snage današnjih višejezgrenih računalnih procesora.

Pojedini dijelovi programskog rješenja izdvojeni su u posebne potprojekte u skladu s uvriježenim praksama struke radi lakšeg izdvajanja određenih funkcionalnosti i korištenja u drugim primjenama. U sljedećim potpoglavljima ulazi se u više detalja o izvebi pojedinih dijelova sustava.

[*].NET Framework verzija libSVM-a je dostupna na web-adresi: http://www.matthewajohnson.org/software/svm.html

6.1. Pomoćni modul

Pomoćni modul naziva TakeLab.Common je mjesto na kojem je definirano nekoliko stvari koje se koriste u ostalim modulima.

Tu je definirano sučelje IContentExtractor koje implementiraju svi ekstraktori. Na taj se način ostale module u programu izolira od poznavanja točne metode koja je u određenom trenutku u upotrebi, što omogućuje lakšu promjenu metoda, te također omogućuje da isti dijelovi programskog koda mogu na jednak način raditi sa bilo kojom od metoda, ovisno o trenutnoj potrebi.

U ovom je modulu definirano i sučelje IMessageLog i klasa MessageLog koja se koristi u vođenju dnevnika izvođenja operacija. Dnevnik izvođenja se prikazuje na grafičkom sučelju aplikacije kako bi korisnik dobio uvid u uspješnost operacija i eventualne probleme koji su se dogodili u postupku.

6.2. Modul za evaluaciju

U ovom su modulu, naziva TakeLab.ExtractEvaluation, implementirane sve funkcionalnosti potrebne za izračun i obradu rezultata ekstrakcije tekstnog sadržaja. Sastoji se od tri klase: Evaluator, EvaluatorResult i Preformatter.

Preformatter klasa u sebi sadrži metode kojima se služimo za početnu obradu ekstrahiranog teksta i zlatnog standarda prije nego se krene u izračun duljine najduljeg zajedničkog podniza između njih. Glavna zadaća joj je dakle, u skladu s već danim opisom postupka, iz ulaznog teksta generirati listu odvojenih riječi od kojih se tekst sastoji. Sadrži metode za:

- prepoznavanje znakova praznine,
- uklanjanje zaostalih HTML-oznaka iz teksta i
- stvaranje liste riječi na temelju teksta.

Klasa EvaluatorResult služi za pohranu ocjene jednog rezultata ekstrakcije, ili za pohranu ukupnog rezultata neke metode. Ona u sebi minimalno sadrži sljedeće podatke:

- duljinu ekstrahiranog teksta,
- duljinu zlatnog standarda i
- duljinu najduljeg zajedničkog podniza tih dvaju tekstova.

Sve tri veličine izražene su kao broj riječi.

Klasa EvaluatorResult sposobna je na temelju tih triju osnovnih podataka izračunati sljedeće mjere:

- preciznost,
- odaziv i
- F1 ocjenu.

Jednom kad se neka od mjera izračuna, rezultat se pohranjuje kako se izračun ne bi morao ponavljati.

Ova klasa također omogućuje izračun skupnog rezultata. Iz liste pojedinačnih instanci objekata klase EvaluatorResult skupni se rezultat računa na način da se pozbrajaju vrijednosti duljina ekstrahiranog teksta, zlatnog standarda i njihove najduljeg zajedničkog podniza, te se stvori nova instanca koja će držati ta tri broja i koja omogućuje računanje ukupnih ocjena.

Pored ovoga, radi potreba ostalih dijelova sustava, ova klasa može nositi i informaciju o imenu datoteke na koju se rezultat odnosi. Također podržava i pohranjivanje podataka u niz znakova u obliku pogodnom za spremanje u tekstnu datoteku formata CSV (engl. *Comma Separated Values*, u prijevodu: vrijednosti odvojene zarezom – iako se u praksi koristi i točka-zarez) koju se može otvoriti u gotovo svim današnjim tabličnim kalkulatorima poput programa Microsoft Excel ili LibreOfice/OpenOffice Calc.

Posljednja od tri klase ovog modula je klasa Evaluator. Ona implementira ključni dio postupka vrednovanja, a to je algoritam traženja duljine najduljeg zajedničkog podniza dvaju zadanih nizova znakova, konkretno između ekstrahiranog teksta i teksta zlatnog standarda. Na ulazu prima dva niza znakova koji sadrže cijeli tekst dvaju ulaznih datoteka, a izlaz joj je objekt klase EvaluatorResult.

6.3. Modul za ekstrakciju

U modulu za ekstrakciju, naziva TakeLab.Extraction, izvedene su klase koje omogućuju pokretanje određenih metoda ekstrakcije. Sve klase koje omogućuju ekstrakciju tekstnog sadržaja implementiraju sučelje IContentExtractor definirano u pomoćnom modulu.

Modul posjeduje klase koje omogućuju pokretanje četiri algoritma ekstrakcije:

- Readability, u klasi istog naziva,
- Boilerpipe Article Extractor, u klasi BoilerpipeAE,
- Boilerpipe Default Extractor, u klasi BoilerpipeDE i
- Body Text Extraction, u klasi BodyTextExtraction.

Od četiri navedena algoritma, jedino je Body Text Extraction doista i implementiran u ovom modulu, dok se ostale tri klase oslanjaju na vanjske module koji obavljaju taj zadatak. Neće se detaljnije ulaziti u implementaciju tih algoritama.

U sljedećim potpoglavljima opisana je implementacija algoritma Body Text Extraction, zatim implementaciju vektora značajki koji se koristi kao ulaz u hibridnim ekstraktorima, implementacije hibridnih ekstraktora i implementacije segmentnog ekstraktora i ekstraktora po predlošku.

6.3.1. Algoritam Body Text Extraction

Body Text Extraction implementiran je uz pomoć programske biblioteke Html Agility Pack. To je biblioteka otvorenog koda koja omogućuje brzo i pouzdano analiziranje strukture HTML-dokumenata, i sposobna je podnijeti greške u strukturi dokumenta kakve se znaju naći na webu. Pomoću nje se u prvom predkoraku ulazni HTML-dokument pretvara u DOM-stablo.

Potom se iz DOM-stabla uklanjaju svi elementi "script" i "style", pošto se ispod njih nalaze segmenti HTML-dokumenta koje Html Agility Pack deklarira kao tekstne elemente, što oni u osnovi i jesu, ali se tu konkretno radi o programskom kodu – unutar oznake "script" obično se nalazi kod u JavaScriptu koji upravlja interaktivnim ponašanjem stranice, dok se unutar oznake "style" nalazi CSS-kod koji oblikuje njen izgled. Taj se kod nipošto ne smije naći u izlazu, a pošto zna biti i poveći, ako ostane značajno će utjecati na rad algoritma.

Nakon toga se gradi posebna lista tokena, po načelu algoritma BTE, prolaženjem kroz cijelo DOM-stablo dokumenta. Svaki token proizvedene liste ima svoju vrstu, a to može biti oznaka ili riječ. Kako bi se smanjio prostor u kojem je potrebno vršiti pretraživanje, grupiramo susjedne tokene u jedan ovakav element. Svaki element koji označava skupinu uzastopnih oznaka onda sadrži i broj ozaka koje su u njemu sadržane. Ako se naiđe na blok teksta, koji je u DOM-stablu predstavljen s jedinstvenim listom koji sadrži cijeli blok, dodaje se novi element u listu, označava da se radi o tekstu, i u brojčanu vrijednost tog elementa upisuje broj riječi u bloku.

Smatra se da su riječi razdvojene znakovima praznine. Svaki će tekstni element u ovoj listi sadržavati i referencu na izvorni DOM-element iz kojeg je stvoren kako bi se kasnije moglo doći do tog teksta ukoliko ga algoritam ocijeni korisnim.

Potom se primjenjuje sljedeći algoritam:

Algoritam Body Text Extraction

Ulaz: *B*: pripremljena lista; *N*: duljina pripremljene liste
Izlaz: *maxVrijedI*, *maxVrijedJ*: indeksi u listi *B*

```
ukupnoOznaka ← suma vrijednosti u svim oznaka elementima liste B
maxVrijed ← ukupnoOznaka
maxVrijedI ← -1
maxVrijedJ ← -1
za (i ← 1 .. N)
{
        trenutnoZaI ← ukupnoOznaka
        za (j ← i .. N)
        {
                ako (B[j] je tekstni element)
                {
                        trenutnoZaI ← trenutnoZaI + ( vrijednost u B[j] )
                }
                inače
                {
                        trenutnoZaI ← trenutnoZaI - ( vrijednost u B[j] )
                }

                ako (trenutnoZaI > maxVrijed)
                {
                        maxVrijed ← trenutnoZaI
                        maxVrijedI ← i
                        maxVrijedJ ← j
                }
        }
}
```

Konačan rezultat ovog algoritma su dva indeksa. Ukoliko su oba -1, rezultat ekstrakcije BTE algoritmom je prazan niz. Inače, uzima se vrijednosti tih indeksa, prolazi kroz listu tokena i uzima sve tekstne članove između ta dva indeksa

(uključivo). Ti nizovi se spajaju, i to na način da se dodatno između dva bloka ubacuje jedan znak razmaka, ali samo ukoliko niti prethodni blok već završava, niti sljedeći već počinje s nekim znakom praznine.

Na ovaj način se dobiva tekst koji, po algoritmu BTE, čini koristan tekst početnog HTML-dokumenta.

6.3.2. Vektor značajki

Kod za generiranje vektora značajki je implementiran u klasi FeatureVector. Značajke koje se generiraju opisane su u poglavlju o postupku. Generirani vektor značajki sprema se u polje brojeva s pomičnim zarezom.

6.3.3. Prva varijanta hibridnog ekstraktora

Prva varijanta hibridnog ekstraktora implementirana je u dvije klase. Klasa SVMHybridFullDocBased služi za čuvanje gotovog istreniranog modela i ima mogućnosti spremanja modela na disk i njegovog kasnijeg učitavanja s diska. Ona je izvedena iz klase TrainedHybrid koja sadrži neke zajedničke funkcionalnosti i predstavlja zajedničko sučelje varijanti hibridnih ekstraktora, koje je bilo potrebno za lakše upravljanje njima kroz grafičko sučelje programa.

Klasa SVMHybridFullDocBased nudi metodu za davanje predviđanja. Prima vektor značajki HTML-dokumenta, a kao rezultat vraća broj koji označava metodu ekstrakcije za koju predviđa da će na tom dokumentu ostvariti najbolji rezultat.

Najvažniji dio implementacije ove varijante hibridnog ekstraktora je klasa koja upravlja učenjem SVM modela, klasa SVMTrainingFullDocBased. Da bi se pokrenulo učenje, potrebno je sakupiti rezultate evaluacije osnovne četiri metode nad skupom za učenje. Ova klasa potom priprema vektore značajki svih HTML-dokumenata u skupu i povezuje ih s brojem koji označava metodu koja je na tom dokumentu ostvarila najbolji F1-rezultat. S ovako pripremljenim skupom podataka trenira se SVM-klasifikator. Rezultat je instanca klase SVMHybridFullDocBased.

6.3.4. Segmentacija HTML-dokumenta

Jedna od osnovnih predzadaća druge varijante hibridnog ekstraktora jest rastaviti HTML dokument na smislene segmente. Cilj tog rastavljanja je postići takvu segmentaciju gdje bi se, idealno, u pojedinim segmentima nalazio samo zanimljiv ili samo nezanimljiv tekst, pošto se odluka o ekstrakciji donosi za cijeli segment.

Ova funkcionalnost implementirana je u klasi SVMHybirdSectionBased.

Prije početka postupka iz HTML-dokumenta uklanjaju se oznake "script" i "style", kao i kod algoritma Body Text Extraction, jer ne sadrže koristan tekst, već kod.

Početni korak postupka jest pronalaženje "body" HTML-oznake, ukoliko je prisutna u dokumentu. Ništa izvan te oznake u konačnici se ne prikazuje čitatelju, pa stoga ne može biti dio zanimljivog teksta. Ukoliko oznaka "body" nije prisutna promatra se cijeli dokument.

U traženju strukturnih segmenata HTML-dokumenta fokus je na oznakama "div" i "table" jer su to oznake koje se u praksi koriste za razdvajanje dijelova HTML-dokumenta i pomoću njih se obično postiže oblikovanje strukture izgleda kod prikaza web-stranice.

Postupak se sastoji od dva algoritma koji se pozivaju rekurzivno – algoritam analize segmenta, i algoritam analize tablice. Algoritam analize segmenta pokreće se prvi i to nad oznakom "body" ili cijelim dokumentom ako oznaka "body" nije prisutna. Pored reference na DOM-čvor koji analizira, algoritam dobiva i podatak o ukupnoj duljini teksta koji se prenosi nepromijenjen kroz sve razine rekurzije i služi za sprječavanje presitne segmentacije dokumenta. Algoritmi ulaze u dublju analizu podčvorova samo ako se u njima nalazi više od polovine teksta koji se nalazi unutar oznake "body", ili cijelog dokumenta ako ta oznaka nije definirana.

Algoritam analize segmenta

Ulaz: *vrh*: čvor DOM-stabla; *ukDuljina*: cijeli broj, ukupna duljina teksta u cijelom dokumentu

Izlaz: *rezultat*: lista segmenata, svaki segment je niz znakova

```
rezultat ← prazna lista
ako (vrh nema djecu)
{
        kraj
}
segmentniSpremnik ← inicijaliziraj tekstni spremnik
za (dijete ← svako dijete čvora vrh osim HTML-komentara)
{
        ako (dijete je HTML-oznaka i dijete sadrži InnerText* dulji od (ukDuljina / 2)
        i dijete ima svoju djecu)
        {
                ako (segmentniSpremnik sadrži tekst)
                {
                        dodaj u rezultat: sadržaj od segmentniSpremnik
                        isprazni segmentniSpremnik
                }
                ako (dijete je oznaka "table")
                {
                        dodaj u rezultat: pozovi analizu tablice (dijete, ukDuljina)
                }
                inače
                {
                        dodaj u rezultat: pozovi analizu segmenta (dijete, ukDuljina)
                }
        }
        inače ako (dijete nije oznaka "div" i dijete nije oznaka "table")
        {
                ako (OuterHtml** od dijete ne sadrži samo znakove praznine)
                {
                        dodaj u segmentniSpremnik: OuterHtml od dijete
                }
        }
```

*Polje *InnerText* je definirano standardom za HTML DOM-model i sadrži ukupan tekst ispod nekog DOM-čvora, bez oznaka.

**Polje *OuterHtml* također je definirano standardom HTML DOM-modela i sadrži izvorni tekstni zapis HTML-čvora s cijelim sadržajem.

Algoritam analize segmenta

```
        inače
        {
                ako (segmentniSpremnik sadrži tekst)
                {
                        dodaj u rezultat: sadržaj od segmentniSpremnik
                        isprazni segmentniSpremnik
                }
                dodaj u rezultat: OuterHtml od dijete
        }
}
ako (segmentniSpremnik sadrži tekst)
{
        dodaj u rezultat: sadržaj od segmentniSpremnik
        isprazni segmentniSpremnik
}
```

Algoritam analize tablice

Ulaz: *tablica*: DOM-čvor oznake "table"; *ukDuljina*: cijeli broj, ukupna duljina teksta u cijelom dokumentu

Izlaz: *rezultat*: lista segmenata, svaki segment je niz znakova

```
rezultat ← prazna lista
ako (vrh nema djecu)
{
        kraj
}
dijete ← prvo dijete od tablica
ponavljaj
{
        ako (dijete je oznaka i (dijete je oznaka "td" ili dijete je oznaka "caption"))
        {
                ako (dijete sadrži InnerText dulji od (ukDuljina / 2))
                {
                        dodaj u rezultat: pozovi analizu segmenta(dijete, ukDuljina)
                }
                inače
                {
                        dodaj u rezultat: OuterHtml od dijete
                }
        }
        inače ako (dijete ima djecu)
        {
                dijete ← prvo dijete od dijete
                sljedeća iteracija
        }

        dok (dijete je zadnje dijete svojeg roditelja i dijete ≠ tablica)
        {
                dijete ← roditelj od dijete
        }
        ako (dijete = tablica)
        {
                prekid petlje
        }
        inače
        {
                dijete ← sljedeće dijete istog roditelja
        }
```

Algoritam analize tablice
}

6.3.5. Druga varijanta hibridnog ekstraktora

Središnja klasa ove varijante hibridnog esktraktora je klasa SVMHybridSectionBased. Ona u sebi drži naučeni model SVM-klasifikatora i nudi osnovne funkcionalnosti poput spremanja modela na disk i ponovnog učitavanja s diska.

Osim što je izvedena iz apstraktne klase TrainedHybrid, kao i klasa SVMHybridFullDocBased, ova klasa dodatno implementira sučelje IContentExtractor, koje definira pristup do metode za ekstrakciju kakvu implementiraju i osnovni ekstraktori. Razlog ovoj razlici između implementacije ove varijante i prve varijante hibridnog ekstraktora je u razlici u njihovom pristupu radu. Dok je izlaz prve varijante samo broj koji određuje koju metodu potom treba primijeniti na HTML-dokumentu, izlaz druge varijante je već ekstrahirani tekst pa se ona može koristiti kao samostalni ekstraktor.

Ova klasa sadrži sav kod koji obavlja pripremu ulaznih podataka za SVM-klasifikator.

Klasa sadrži metodu koja priprema podatke za učenje. Ona prima jedan Stream (standardna .NET klasa za pristup do raznih izvora podataka, primjerice do datoteke na disku) koji sadrži HTML-dokument, zatim prima i sav tekst zlatnog standarda, te prag ekstrakcije – koeficijent koji određuje kolika mora biti mjera zastupljenosti teksta u segmentu u tekstu zlatnog standarda da bi se tekst smatrao zanimljivim. Njezin rezultat je niz vektora značajki, po jedan za svaki segment, te odgovarajući niz predviđanja, po jedno za svaki segment. Svaki vektor značajki, pored značajki samog segmenta, sadrži i mjere zastupljenosti segmenta u ekstrahiranim tekstovima dobivenim primjenom osnovnih algoritama na ulazni HTML-dokument.

Metoda za ekstrakciju prima samo Stream koji sadrži HTML-dokument, razdvaja ga na segmente i za svaki segment generira vektor značajki jednako kao kod učenja. Vektori se predaju naučenom SVM modelu koji vrši klasifikaciju u zanimljive i nezanimljive. Iz segmenata koji odgovaraju vektorima koji su klasificirani kao zanimljivi izvlači se ukupan tekstni sadržaj koji se vraća kao rezultat metode.

6.3.6. Segmentni ekstraktor

Segmentni ekstraktor izveden je samo kao privremena modifikacija druge varijante hibridnog ekstraktora za potrebe ispitivanja i nema trajnu implementaciju unutar programa, no ista se može vrlo lako izvesti ukoliko se za tim pokaže potreba.

6.3.7. Ekstraktor po predlošku

Ekstraktor koji radi po predlošku izveden je u klasi TemplateExtractor. Ova klasa također implementira sučelje IContentExtractor. Njena zadaća je dvojaka – analiziranje predloška te primjena predloška u ekstrakciji.

Datoteka predloška promatra se redak po redak teksta. Svaki prazan redak se preskače. U ostalim retcima prvo se traži postojanje znaka jednakosti ("="). Ukoliko se taj znak ne pronađe, redak predstavlja selekciju po elementu. Ukoliko se pronađe, onda se provjerava nalazi li se iza znaka jednakosti ikakav tekst. Ako da, radi se o selekciji elemnta po atributu sa zadanom vrijednosti, a ako ne, o selekciji elementa po atributu. Pojedini se dijelovi selekcijskih izraza izvlače i gradi se lista selektora koja sadrži sve podatke u odvojenom obliku, pogodnijem za korištenje u ostatku postupka.

Svaki se selektor dodatno provjerava. Smatra se greškom ako neko ime, bilo HTML-elementa ili atributa, sadrži išta osim slova, brojki, i znakova crtice ("-") i podvlake ("_"). Vrijednosti atributa posebno se ne provjeravaju, osim što se inzistira da jedna vrijednost u sebi ne smije istovremeno sadržavati i apostrofe i navodnike, što je potrebno zbog načina na koji se te vrijednosti koriste. Provjere su mogle biti i preciznije, no to nije potrebno, jer će kasniji postupak naići na grešku ukoliko ime ili vrijednost ne zadovoljavaju uvjete HTML-standarda.

Iz dobivene liste selektora potom se gradi XPath izraz. XPath je formalni jezik kojim se definiraju upiti s kojima se može pretraživati XML-dokumente. Biblioteka Html Agility Pack podržava da se XPath izrazima pretražuje DOM-stablo HTML-dokumenta.

S dobivenim XPath izrazom vrši se pretraživanje po DOM-stablu. Rezultat je lista svih HTML-elemenata (u obliku DOM-čvorova) koji udovoljavaju barem jednom od zadanih uvjeta. Potom se iz svakog dobivenog elementa, ukoliko se utvrdi da on nije već sadržan u nekom od drugih selektiranih elemenata, izvlači sav tekstni sadržaj. Ti se sadržaji spajaju u izlazni ekstrahirani tekst na način da se dodatno između

svaka dva tekstna segmenta dodaje znak za prelazak u novi red, radi bolje čitljivosti rezultata.

6.4. Glavni modul

Glavni modul programa je modul TakeLab.Crawler. U njemu je definirano grafičko sučelje programa te je tu implementirana sva njegova interakcija s korisnikom i s ostalim modulima. Svaki od postupaka koji se mogu raditi s ovim programom ovdje ima implementiranu klasu koja upravlja tim postupkom.

- Klasa EvaluationProcess upravlja postupkom vrednovanja;
- Klasa ExtractionProcess upravlja postupkom ekstrakcije korištenjem jednog od četiri osnovna algoritma;
- Klase ATypeTrainingProcess i ATypeExtractionProcess upravljaju učenjem prve varijante hibridnog ekstraktora i ekstrakcijom pomoću njega;
- Klase BTypeTrainingProcess i BTypeExtractionProcess upravljaju učenjem druge varijante hibridnog ekstraktora i ekstrakcijom pomoću njega;

Dio ovih klasa koriste Task.Parallel biblioteku kako bi ubrzale svoj rad.

Svaki od procesa ekstrakcije drži svoje rezultate u posebnom direktoriju. Kroz grafičko sučelje korisnik definira osnovni direktorij pohrane koji se proslijeđuje svim procesima koji upravljaju ekstrakcijom. Očekuje se da će se izvorni HTML-dokumenti nalaziti u poddirektoriju "html" od direktorija pohrane, dok se zlatni standard nalazi u poddirektoriju "clean".

Svaka metoda ekstrakcije stvara svoj poddirektorij koji naziva po imenu svojeg ekstraktora, koje u slučaju hibridnih ekstraktora sadrži i datum i vrijeme kad je esktraktor treniran, dok se kod ekstrakcije po predlošku u imenu podirektorija navodi ime datoteke predloška, bez nastavka.

Proces evaluacije pak prolazi kroz sve poddirektorije direktorija pohrane osim "html" i "clean" te evaluira svaku metodu uspoređujući sadržaj njenog direktorija sa sadržajem u direktoriju "clean". U svakom od poddirektorija koje je evaluirao ovaj će proces ostaviti datoteku naziva "evaluation.csv", tekstnu datoteku formata CSV, u kojoj se mogu pronaći svi pojedinačni rezultati evaluacije skupa s ukupnim rezultatom koji bude smješten u prvom retku.

Kroz grafičko sučelje moguće je spremati i ponovno učitavati jednom istrenirane modele hibridnog klasifikatora.

7. Zaključak

U radu je opisana problematika ekstrakcije tekstnog sadržaja iz HTML-dokumenata, opisane su metode za vrednovanje uspješnosti algoritama za ekstrakciju, te je prezentiran presjek postojećih rješenja za ovaj problem.

Razvijene su dvije varijante hibridnog ekstraktora koje pokušavaju postići bolji rezultat kombiniranjem postojećih algoritama pomoću SVM algoritma strojnog učenja. Prva varijanta pokušava predvidjeti, na temelju strukturnih karakteristika HTML-dokumenta, koja će od postojećih metoda postići najbolji rezultat, dok druga rastavlja HTML-dokument na segmente i uz pomoć strukturnih karakteristika segmenta i rezultata postojećih metoda bira koji su segmenti zanimljivi. Razvijena je metoda segmentacije HTML-dokumenata te dodatni segmentni ekstraktor sadržaja koji uči raspoznavati zanimljive od nezanimljivih segmenata isključivo po svojstvima segmenta.

Rezultati su pokazali kako prva varijanta nije bila uspješna. Druga je varijanta pak ostvarila dobar rezultat. Segmentni ekstraktor je također ostvario dobar rezultat, no rezultat druge varijante je ipak bolji pa se može zaključiti kako je informacija o rezultatu postojećih metoda korisna. No nijedna od metoda nije ostvarila bolji rezultat od algoritma BTE.

Poboljšanje se može tražiti u ispitivanju drugačijih ili većeg broja značajki HTML-dokumenta i segmenta. Također, kod druge varijante rezultati upućuju kako bi vrijedilo istražiti moguća poboljšanja u algoritmu segmentiranja ili u načinu mjerenja prisutnosti nekog tekstnog segmenta unutar teksta zlatnog standarda.

8. Literatura

(Alpert i Hajaj, 2008) Alpert J, Hajaj N, We knew the web was big..., objava na blogu, 2008., dostupno na web-adresi: http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

(Bar-Yossef i Rajagopalan, 2002) Bar-Yossef Z, Rajagopalan S, Template Detection via Data Mining and its Applications. In WWW, pp 580–591, 2002.

(Baroni i dr., 2008) Baroni M, Chantree F, Kilgarriff A, Sharoff S, Cleaneval: a competition for cleaning web pages, Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), 2008.

(Cai i dr., 2003) Cai D, Yu S, Wen JR, Ma WY, VIPS: a Vision-based Page Segmentation Algorithm, Microsoft Research, MSR-TR-2003-79, 2003.

(Chang i Lin, 2011) Chang C and Lin C, LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Biblioteka dostupna na web-adresi http://www.csie.ntu.edu.tw/~cjlin/libsvm

(Cortes i Vapnik, 1995) Cortes C, Vapnik VN, Support-Vector Networks, Machine Learning, 20, 1995.

(Finn i dr., 2001) Finn A, Kushmerick N, Smyth B, Fact or Fiction: Content Classification for Digital Libraries, DELOS Workshop: Personalization and Recommender Systems in Digital Libraries, 2001.

(Fletcher, 2011) Fletcher WH, Corpus Analysis of the World Wide Web, 2011., dostupno na web-adresi: http://www.webascorpus.org/

(Gottron, 2007) Gottron T, Evaluating Content Extraction on HTML Documents. In ITA, pp 123–132, 2007.

(Gottron, 2008a) Gottron T, Content Code Blurring: A New Approach to Content Extraction, 19th International Workshop on Database and Expert Systems Applications (DEXA 2008), Turin, Italy, September 2008.

(Gottron, 2008b) Gottron T, Combining content extraction heuristics: the CombinE system, International Conference on Information Integration and Web-based Applications & Services (IIWAS), pp 591-595, 2008.

(Hirschberg, 1975) Hirschberg DS, A linear space algorithm for computing maximal common subsequences, *Commun. ACM* 18, 6, July 1975, pp 341-343.

(Kohlschütter i dr., 2010) Kohlschütter C, Fankhauser P, Nejdl W, Boilerplate Detection using Shallow Text Features, WSDM'10, New York City, New York, USA, February, 2010.

(Kovačić, 2011) Kovačić T, Evaluation Metrics for Text Extraction Algorithms, objava na blogu, 2011., dostupno na web-adresi: http://tomazkovacic.com/blog/74/evaluation-metrics-for-text-extraction-algorithms/

(Krišto, 2009) Krišto I, Postupak čišćenja Web-stranica u svrhu dubinske analize teksta, Završni rad, Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, Zagreb, 2009.

(Levenshtein, 1966) Levenshtein V, Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, Soviet Physics-Doklady, 10, vol. 10, pp. 707-710, 1966.

(Lin i Ho, 2002) Lin SH, Ho JM, Discovering Informative Content Blocks from Web Documents, SIGKDD '02, Edmonton, Alberta, Canada, July, 2002.

(Ljubešić i Erjavec, 2011) Ljubešić N, Erjavec T, hrWaC and slWac: Compiling Web Corpora for Croatian and Slovene, Text, Speech and Dialogue 2011. Lecture Notes in Computer Science, Springer, 2011.

(Mantratzis i dr., 2005) Mantratzis C, Orgun MA, Cassidy S, Separating XHTML Content from Navigation Clutter Using DOM-structure Block Analysis. In S. Reich and M. Tzagarakis, editors, Hypertext, pp 145–147, ACM, 2005.

(Marek i dr., 2007) Marek M, Pecina P, Spousta M, Web Page Cleaning with Conditional Random Fields, In Proceedings of the 3rd Web As a Corpus Workshop, Incorporating CLEANEVAL, pp 155-162, Louvain-la-Neuve, Belgium, 2007.

(Pasternack i Roth, 2009) Pasternack J, Roth D, Extracting Article Text from the Web with Maximum Subsequence Segmentation, WWW 2009, Madrid, Spain

(Pinto i dr., 2002) Pinto D, Branstein M, Coleman R, Croft WB, King M, Li W, Wei X, QuASM: A System for Question Answering Using Semi-Structured Data, JCDL'02, July 13-17, 2002, Portland, Oregon, USA

(Pomikálek, 2011) Pomikálek J, Removing Boilerplate and Duplicate Content from Web Corpora, Ph.D. thesis, Masaryk University, Faculty of Informatics, Brno, 2011.

(Ratcliff i Metzener, 1988) Ratcliff JW, Metzener D, Pattern Matching: The Gestalt Approach, Dr. Dobb's Journal, pp 46, July 1988.

(Weninger i dr., 2010) Weninger T, Hsu WH, Han J, CETR - Content Extraction via Tag Ratios, WWW 2010, Raleigh, North Carolina, USA, 2010., ACM 978-1-60558-799-8/10/04.

(Weninger i Hsu, 2008) Weninger T, Hsu WH, Text Extraction from the Web via Text-to-Tag Ratio, Workshop on Text Info. Ret. at Int. Conf. on Data. and Expert Sys. (TIR'08), Turin, Italy, September 2008.

(Yi i dr., 2003) Yi L, Liu B, Li X, Eliminating Noisy Information in Web Pages for Data Mining, SIGKDD '03, Washington, DC, USA, August, 2003.

Sažetak: Opisana je problematika ekstrakcije tekstnog sadržaja iz HTML-dokumenata, tj. uklanjanja nebitnog sadržaja. Predstavljene su metode za vrednovanje uspješnosti algoritama za ekstrakciju, te je dan presjek postojećih rješenja. Razvijene su dvije varijante hibridnog ekstraktora koje kombiniraju postojeće algoritme pomoću SVM-algoritma strojnog učenja. Razvijene metode su eksperimentalno vrednovane nad skupom dokumenata CleanEval. Rezultati pokazuju da kombinacija nije bila uspješna jer ne nadmašuje rezultat najboljeg postojećeg upotrijebljenog algoritma, BTE. Razvijen je i ekstraktor sadržaja po predlošku.

Ključne riječi: HTML, web-stranice, uklanjanje šuma, ekstrakcija tekstnog sadržaja, hibridni ekstraktor, Body Text Extraction, Boilerpipe, Readability, SVM

Abstract: The problem of extracting textual content, or boilerplate removal, from HTML documents is described. Methods of evaluating results of extraction algorithms are presented, along with an overview of existing solutions. Two variants of a hybrid extractor were developed which combine existing algorithms using the SVM machine learning algorithm. The developed methods were experimentally evaluated on the CleanEval dataset. Results indicate that the combinations were not succesful as they were unable to achieve a better score than the best existing used algorithm, BTE. A template-based content extractor was also developed.

Keywords: HTML, web pages, boilerplate removal, content extraction, hybrid extractor, Body Text Extraction, Boilerpipe, Readability, SVM