SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 851

Postupak čišćenja web stranica u svrhu dubinske analize teksta

Ivan Krišto

Zagreb, lipanj 2009.

Sadržaj

1. Uvod

Metode dubinske analize teksta i pretraživanja informacija oslanjaju se na statističke metode koje za ispravan rezultat moraju biti primjenjene na velikoj količini podataka. Dodatno, suvremeni jezici su dinamični te tekstovni podatci nad kojima se vrši dubinska analiza trebaju odražavati trenutno stanje jezika.

Internet je neiscrpan izvor *svježih* tekstovnih podataka na raznim jezicima koji, osim slabo strukturiranog sadržaja koji često ne poštuje gramatička i pravopisna pravila (npr. poruke na forumima, komentari korisnika i sl.), sadrži novinske arhive, stara i nova književna djela, tekstove iz raznih područja ljudskog istraživanja pisane znanstvenim stilom i sl.

Problem internetskog sadržaja jest multimedijalnost. Uz primarni sadržaj (onaj koji sadrži smislene informacije zbog kojih stranica postoji) nalazi se i *sekundarni sadržaj*, tj. elementi navigacije, opis stranice, a često i reklame. Liu (2004) navodi širi opis navedenog problema.

Kilgarriff and Grefenstette (2003); Baroni and Ueyama (2006) kao prednosti korištenja *World Wide Weba* za potrebe dubinske analize navode veličinu,[1] stalni rast, brzinu izgradnje skupa podataka, nisku cijenu te raznolikost sadržaja koja uključuje sadržaj koji se ne može naći u tradicionalnim medijima (npr. komentari čitatelja). Kao nedostatak naveli su veliku količinu šuma (npr. dijelovi navigacije), dupliciran sadržaj, manja količina lektoriranja pri objavljivanju sadržaja i slabu mogućnost kontrole sadržaja. Liu (2004) sistematizira navedene probleme i prednosti korištenja weba kao izvora informacija te donosi pregled radova koji se bave navedenim područjem.

Pri dubinskoj analizi teksta sekundarni sadržaj tvori nezanemariv šum, te je potrebno provesti postupak čišćenja radi uklanjanja šuma. Postupak čišćenja otežava problem sličnosti primarnog i sekundarnog sadržaja.

U 2. poglavlju navodi se pregled dosad razvijenih metoda čišćenja web stranica

[1] Npr. pokazano je da jednostavni algoritam za razrješavanje višeznačnosti nadmašuje kvalitetniji algoritam ako je treniran na većoj količini podataka (Banko and Brill, 2001)

i izlučivanja sadržaja, komentira uspješnost i specifični problemi te vrši usporedba. Sličan pregled radova se može naći u (Liu, 2004). U 3. poglavlju iznosi se kritika weba kao izvora podataka za dubinsku analizu teksta. U 4. poglavlju navode se metode razvijene u sklopu rada, usporedbu s nekim metodama opisanim u 2. poglavlju te prikaz rezultata rada. Opis evaluacije postupka dat je u 5. poglavlju, a opis implementacije metoda i evaluacije u 6. poglavlju. Rad završava zaključkom i smjernicama za daljni razvoj.

2. Srodna rješenja

Informacije s weba možemo dohvaćati u više oblika, npr. kao *nestrukturirani* ili *strukturirani* tekst. Pojam *strukturirani* tekst odnosi se na organizaciju teksta pri kojoj su različiti dijelovi odvojeni strukturnim elementima (kod HTML–a[1] to je odvajanje različitih blokova stranice). Način razlikovanja dijelova obično se odnosi na semantičku razliku.

Za potrebe dubinske analize često nam je dovoljan nestrukturirani tekst, te možemo iskoristiti razne alate za pretraživanje weba (npr. Google, AltaVista i sl.). Kilgarriff and Grefenstette (2003) naveli su primjer korištenja tražilica i iznesena usporedba informacija dobivenih obradom podataka skupljenih pretraživanjem weba pomoću tražilica i pretraživanjem korpusa. Nedostatci takvog pristupa su ograničenost konteksta i velika količina *šuma*.

Navedene nedostatke može riješiti dohvaćanje informacija kao strukturiranog teksta. Time gubimo količinu podataka koje nam tražilice nude, ali dobivamo potpuni kontekst podatka (tražilice također dohvaćaju strukturirani tekst te ga indeksiraju, ali za te potrebe koriste ogromnu računalnu moć). Informacije s weba izvorno se dohvaćaju kao strukturirani tekst u HTML formatu. Informacije možemo dohvatiti kao kompletan HTML dokument te dohvatiti potpuni sadržaj dokumenta. No tim pristupom suočavamo se s problemom prijespomenutih negativnih osobina internetskog sadržaja. Metode razvijene u tu svrhu zasnivaju se na otkrivanju mjesta primarnog sadržaja u objektnom modelu dokumenta[2] stranice ili uklanjanju sekundarnog sadržaja.

Kushmerick (1999) navodi primjer uklanjanja sekundarnog sadržaja uklanjanjem slikovnih reklamnih poruka. Metoda se temelji na pravilima dobivenim analizom HTML koda slikovnih reklama (npr. karakteristična veličina i širina, alternativni tekst `img` elementa i sl.).

Metode za određivanje primarnog sadržaja obično se temelje na izradi pred-

[1]URL: http://www.w3.org/TR/html401/
[2]URL: http://www.w3.org/DOM/DOMTR

loška (skup pravila) za pojedino web sjedište (engl. *web site*) koji opisuje poziciju primarnog sadržaja (jer se HTML stranice web sjedišta često generiraju na temelju predloška stila te pune podatcima iz baze podataka). Postupak izrade predloška možemo izvesti u potpunosti ručno (što nema praktičnu primjenu zbog količine dokumenata) ili naglasiti primarni sadržaj u nekoliko reprezentativnih web stranica tog sjedišta te postupkom strojnog učenja izgraditi uzorak. Cilj koji se želi postići jest potpuna automatizacija određivanja primarnog sadržaja.

Pri odabiru metode bitno je odrediti kakav skup stranica se želi obraditi: radi li se samo nad web sjedištem (što je uglavnom slučaj) ili je potrebno obrađivati samostalne HTML stranice. Obrada samostalnih stranica manje je zastupljena zbog slabe mogućnosti korištenja algoritama učenja. Ovaj rad orijentiran je na obradu samostalnih stranica.

Bar-Yossef and Rajagopalan (2002) navode problem određivanja predloška i praktično rješenje temeljeno na frekvenciji pojavljivanja skupa elemenata stranice. Sličnu metodu namjenjenu za obradu web sjedišta razvili su Yi et al. (2003). Njihova se metoda temelji na izgradnji *stabla stila* (engl. *style tree*), pri čemu se uzima u obzir broj pojavljivanja pojedine grupe elemenata u datom skupu stranica. Metoda se pokazala uspješnija od metode temeljene na frekvenciji pojavljivanja skupa elemenata stranice.

Pristup gradnji predloška često se temelji na načinu izrade web stranica. Baroni and Ueyama (2006) opisuju postupak koji koristi heuristiku koju su razvili Finn et al. (2001), a temelji se na pretpostavci da je u dijelovima bogatim primarnim sadržajem koncentracija HTML elemenata manja od koncentracije u dijelovima bogatim sekundarnim sadržajem (Arias et al. (2009) pokazali su da se navedena pretpostavka često krši, te su opisali kako ta činjenica utječe na izvlačenje primarnog sadržaja). Dana je usporedba izgrađenog korpusa s postojećim korpusom izgrađenom na temelju novinske arhive.

Traženje predloška može se izvesti pristupom koji u obzir uzima način izgradnje web stranica bogatih sadržajem. Kod stranica bogatih sadržajem sekundarni i primarni sadržaji obično su odvojeni u vizualno zasebne blokove da bi se korisnik što lakše snašao. Song et al. (2004) opisuju model određivanja značaja blokova dobivenih algoritmom segmentacije stranice na temelju vizualne strukture (*VIPS*: VIsion-based Page Segmentation algorithm) opisanom u (Cai et al., 2003). Sličan pristup opisali su Kovačević et al. (2002).

Arias et al. (2009) razvili su metodu koja se zasniva na sljedećim pretpostavkama: (1) primarni sadržaj od sekundarnog je odvojen barem jednim HTML

elementom: (2) unutar primarnog sadržaja ne nalazi se sekundarni, odnosno, primarni sadržaj je cjelovit i neispresjecan te (3) primarni sadržaj ima manju koncentraciju HTML elemenata nego sekundarni. Metoda ima nisku učinkovitost kod stranica kod kojih je primarni sadržaj odjeljen u više kategorija (npr. forumi i blogovi). Naveden je loš rezultat pri izvlačenju sadržaja sa *slashdot* stranica[3] kod kojih komentari posjetitelja često čine veću količinu teksta nego glavni dio, a komentari i glavna vijest strukturno su odvojeni dijelovi stranice. Zbog navedenog, opisana metoda nije primjerena za stranice stvorene od strane samih korisnika (npr. forumi).

[3]URL: http://slashdot.org/

3. Izgradnja korpusa pomoću weba

Kako bi korpus kao zbirka teksta bi bio koristan pri izvlačenju lingvističkog znanja mora sadržavati raspodjelu riječi koja odgovara stvarnom stanju jezika, tj. ne smije imati umjetno uvećan broj pojavljivanja neke riječi (npr. za web stranice karakteristične riječi su `login`, `search`, `help`, i sl. te je njihova frekvencija pojavljivanja unutar domene web dokumenata daleko veća nego u prirodnom jeziku).

Zbirka web sadržaja može se smatrati svojstvenim korpusom. Problem nastaje pri primjeni takvog korpusa zbog velike količine šuma. Argument za korištenje weba kao korpusa je u njegovoj veličini jer se šum javlja u manjem postotku a na raspolaganju imamo enormnu količinu podataka. Ilustracija količine podataka je rezultat koji dobivamo Googleom za upite *cvijeće* i *cvjeće*. Za pojam *cvijeće* Google je pronašao približno 1.160.000 stranica, a za *cvjeće* približno 60.500.[1] Kilgarriff and Grefenstette (2001) naveli su odnos weba i kontrolirano izgrađenih korpusa te diskutirali o problemu prikladnosti weba za uporabu u dubinskoj analizi teksta.

Uz efikasan alat za uklanjanje šuma jednostavno, jeftino i potpuno automatizirano se mogu izgraditi korpusi od više milijuna riječi. Banko and Brill (2001) pokazali su da veličina korpusa utječe na postotak grešaka alata za dubinsku analizu teksta treniranih pomoću tog korpusa.

Problemi korpusa sastavljenih od sadržaja s weba i izradu takvog korpusa opisali su Boleda et al. (2006). Istaknuti problemi su sekundarni sadržaj, gramatičke greške (sadržaj koji je namijenjen objavljivanju na webu u manjoj mjeri se uređuje od sadržaja namijenjenog papirnatoj objavi), višejezičnost (korpus se gradi u svrhu izučavanja određenog jezika pa prisutnost drugog jezika predstavlja šum) i duplikati (česta pojava na webu koja utječe na frekvenciju pojavljivanja riječi).

[1] Upit je izvršen 26. svibnja, 2009.

4. Razvijene metode čišćenja

Čišćenje HTML stranica odnosi se na ekstrakciju primarnog sadržaja. Nakon što se stranice očiste, ukupan sadržaj možemo lako nadodati korpusu. U nastavku su opisane metode razvijene u sklopu ovog rada, navedena uspješnost čišćenja te je komentirana učinkovitost, moguće primjene te uočeni problemi.

4.1. Čišćenje po karakterističnim atributima

HTML stranice grade se od elemenata, a svaki element može imati proizvoljne atribute.

Na osnovu karakterističnih osobina testnih podataka za evaluaciju (opisano u 5.1.) i podataka o automatskom čišćenju iz (Hranj et al., 2009), možemo zapaziti da karakteristične značajke mogu dobro opisati primarni i sekundarni sadržaj (sličnu ideju koristio je Kushmerick (1999) radi uklanjanja slikovnih reklama).

Na temelju navedenog gradimo statističku metodu za izlučivanje sadržaja temeljenu na pravilima opisa atributa sekundarnog sadržaja.

4.1.1. Rezultati analize karakterističnih osobina

Analizom osobina pokazano je da `class` i `id` atributi *ekvivalentno opisuju sadržaj* elementa i da su to zapravo jedini atributi koje moramo razmatrati (`class` i `id` atribute možemo tretirati jednakima, a sve ostale atribute možemo zanemariti). Ovime odbacujemo veliku količinu redundantnih podataka.

Veliki problem čini nepoštivanje semantike elemenata, odnosno činjenica da se sadržaj najčešće nalazi unutar elemenata: `div`, `span`, `p`, `td`, `li`. Elementi `div` i `span` su strukturni (gradivni) elementi, `td` je ćelija unutar tablice koja služi za spremanje tabličnih podataka, `li` stavka liste koja sadrži samo podatak liste, te `p` koji označava paragraf, i kao takav je jedini element među nabrojanima čija je namjena sadržavanje dijela teksta. Po HTML standardu propisanom od strane

World Wide Web Consortium (W3C) organizacije,[1] tekstovni sadržaj trebao bi se nalaziti unutar `p` elemenata, a unutar ostalih elemenata može se nalaziti ako su oni sadržani unutar `p` elementa. Primjer:

```
<P>aaaaaaaaa<DIV>bbbbbbbbb</DIV><DIV>ccccc<P>ccccc</DIV>
```

Slično je s naslovima koji bi se trebali nalaziti isključivo unutar `h1`, `h2` ili `h3` elemenata, ali in nerijetko nalazimo i unutar `div`, `span`, `p` i sl. elemenata.

Sadržaj često možemo prepoznati po atributu koji unutar sebe sadrži riječ koja ga opisuje, npr.: `head`, `headline`, `ahead`, `masthead` i sl. za naslove, dok su riječi unutar atributa koje opisuju sadržaj elementa: `article`, `comment`, `header`, `heading`, `title`, `text`, `content`, `description`, `news`, `story`, `post`. Na stranicama iz hrvatskog govornog područja često se nađu prijevodi navedenih riječi, npr. "tekst – text", "naslov – heading", i sl. Česte su *skraćenice* tih riječi poput: "header – hdr", "text – txt", "description – desc", itd.

Dodatni problem predstavljaju riječi koje istovremeno opisuju i primarni i sekundarni sadržaj, npr. `adcontent`.

Kombinacije više riječi kao atributa su česte i obično su "odvojene" s: '_', '-', velikim/malim slovom (npr. `ad_slug`, `tekst-vijesti`, `bText`).

Na temelju navedenog možemo konstruirati izraze koji opisuju atribute potencijalnog primarnog, odnosno sekundarnog sadržaja. Izrazi se grade karakterističnim riječima i njihovim prefiksima ili sufiksima odvojenim prethodno opisanim načinima.

Lista riječi sadržanih u izrazima koji potencijalno opisuju primarni sadržaj (u obzir se mogu uzeti i prevedene riječi, npr. "article – clanak"): article, content, header, story, body, main, post, title, blog, comment.

Lista riječi sadržanih u izrazima koji potencijalno opisuju sekundarni sadržaj: login, member, tabs, nav, tools, bar, link, ad, search, sponsor, footer, foot, ftr, user, hidden.

4.1.2. Opis postupka

Metoda se temelji na pretpostavci da je sekundarni sadržaj karakterističnije i češće opisan od primarnog.

Umjesto da se pokuša izravno doći do primarnog sadržaja fokus se stavlja na izbacivanje sekundarnog. Ono se vrši izbacivanjem kompletnih elemenata i

[1] URL: http://www.w3.org/

njihovih podčvorova iz DOM-stabla[2] stranice. Cilj nam je dakle izbaciti što više sekundarnog sadržaja. Postupak dodatno možemo proširiti uključivanjem riječi karakterističnih za opisnike primarnog sadržaja. Primjer toga jest pridruživanje težina riječima, te donošenje odluke na osnovu sumarne težine elementa ili cijelog podstabla. Postupak bez korištenja karakterističnih riječi opisnika primarnog sadržaja prikazan je algoritmom 1.

Algorithm 1 Izbacivanje sekundarnog sadržaja po karakterističnim opisnicima

Ulaz: DOM stablo ($DOMStablo$) HTML dokumenta.
Izlaz: DOM stablo bez elemenata koji sadrže sekundarni sadržaj.
for ($E \leftarrow DOMStablo$) **do**
  {$DOMStablo$ obilazimo *preorder* redoslijedom}
  $atrClass \leftarrow class(E)$
  $atrId \leftarrow id(E)$
  for ($s \in Sekundarni$) **do**
    {$Sekundarni$ je skup čestih riječi opisnika sekundarnog sadržaja}
    if ($s \in atrClass \vee s \in atrId$) **then**
      $DOMStablo.ukloni(E)$
    end if
  end for
end for
return $DOMStablo$

4.2. Čišćenje po gustoći elemenata

Baroni and Ueyama (2006) koristi heuristiku temeljenu na strukturi preuzetu od Finn et al. (2001) koja problem izvlačenja sadržaja iz HTML stranica svodi na problem optimizacije, odnosno traženja dijela stranice sa najmanjom gustoćom HTML elemenata. Pretpostavlja se da je sadržaj u dijelu stranice sa manjom gustoćom HTML elemenata.

Pokazano je da navedena metoda nije efikasna za web stranice čiji se sadržaj nalazi unutar više različitih blokova (npr. forumi, blogovi i stranice sa mogučnošću komentiranja). U nastavku je opisana modificirana metoda koja uzima u obzir navedene tipove stranica.

[2]Svaka HTML stranica može se prikazati *Document Object Model* stablom

4.2.1. Opis postupka

Pokušaj poopćenja metode na stranice sa sadržajem u više blokova dovodi do problema definiranja gustoće i određivanja granične gustoće elemenata. Sličan problem se javlja pri vizualnom predstavljanju HTML sadržaja koji su opisali Cai et al. (2003).

Radi određivanja gustoće stranica je predstavljena u obliku stabla čiji su čvorovi HTML elementi, a listovi tekstovni segmenti. Svaki čvor i list sadrže brojčanu informaciju o trenutnoj gustoći koja se računa iterativno pri stvaranju stabla. Postupak se temelji na odvajanju vrijednosti gustoće na negativnu i nenegativnu (granična gustoća je 0^-). Pri dodavanju novog čvora u stablo gustoća se povećava, a pri dodavanju novog tekstovnog segmenta, gustoća se smanjuje s njegovom logaritamskom duljinom. Negativna vrijednost gustoće označava manju koncentraciju elemenata. Postupak pridjeljivanja vrijednosti gustoće opisan je algoritmom 2.

Algorithm 2 Pridjeljivanje gustoće elementima

Ulaz: DOM stablo ($DOMStablo$) HTML dokumenta.
Izlaz: Stranica u obliku stabla čiji čvorovi sadrže informaciju o gustoći.
$g := 0$ {g – trenutna gustoća}
$stablo := \emptyset$
$cvor := stablo$
for ($E \leftarrow DOMStablo$) **do**
  {$DOMStablo$ obilazimo *preorder* redoslijedom}
  if ($cvor = \emptyset \vee E \notin cvor$) **then**
    $cvor := roditelj(E)$ {Tražimo čvor koji sadrži element E}
    $g := 0$
  end if
  if ($E \in Tagovi$) **then**
    $g := g + 1$
  else
    {E je tekstovni segment}
    $g := g - \log_{10} |E|$ {$|E|$ – broj riječi sadržaja elementa E}
  end if
  $cvor.dodaj(E, g)$
end for
return $stablo$

5. Eksperimentalno vrednovanje

Evaluacija se vrši usporedbom nad skupovima ručno i strojno očišćenih dokumenata. Temelj evaluacije je usporedba količine bitnog i nebitnog teksta u strojno očišćenom dokumentu. Za bitni tekst uzima se sve ono što se nalazi unutar ručno očišćenog dokumenta, a za nebitni tekst sadržaj strojno očišćene datoteke koji se ne nalazi unutar ručno očišćene datoteke. Radi usporedbe sadržaj se može razlučiti na riječi ili znakove. Unutar ovog rada odabrana jedinica evaluacije je znak. Na temelju navedenog definiraju se:

TP (*true positive*): jedinice usporedbe koje su bitne i prepoznate su kao bitne.

FP (*false positive*): jedinice usporedbe koje nisu bitne, ali su pogrešno prepoznate kao bitne.

FN (*false negative*): jedinice usporedbe koje su bitne, ali nisu prepoznate kao bitne.

Intuitivno je shvatljivo da ovakva razlučivost može dovesti do netočnih rezultata jer se jedinice usporedbe često ponavljaju te bi na ovaj način nebitan tekst koji u sebi sadrži bitne jedinice pogrešno tretirali kao ispravno dohvaćen. Uz to, dio jedinica može se pojaviti istovremeno i u skupu bitnog teksta i u skupu nebitnog teksta. Zbog navedenih problema određivanje nebitnog teksta vršimo uklanjanjem najvećih zajedničkih podnizova (engl. *longest common subsequence*) jedinica usporedbe iz strojno i ručno obrađenih datoteka. Detaljni postupak prikazan je algoritmom 3.

Mjere koje koristimo su *točnost* i *odziv* rezultata. Manning and Schtze (1999) daju detaljan opis navedenih mjera.

Točnost (engl. *precision*) pokazuje koliko se bitnog teksta nalazi unutar kompletnog sadržaja strojno očišćene datoteke. Određuje se kao:

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

Algorithm 3 Evaluacija rezultata

Ulaz: Dokument očišćen ručno (D_r) te isti dokument očišćen strojno (D_s).

Izlaz: Ispravno prepoznat bitan dio (TP), bitan dio koji nije prepoznat (FN), nebitan dio koji je prepoznat kao bitan (FP).

$TP := \emptyset$, $FP := \emptyset$, $FN := \emptyset$

$NP := D_s \cap^* D_r$ {$\cap^*$ – operator presjeka nije strogo definiran, primjer presjeka je najveći zajednički podniz}

U TP nadodaj NP

Iz D_s izbaci NP

Iz D_r izbaci NP

$FP := D_s$ {Sve što je ostalo u D_s je neispravno proglašeno bitnim}

$FN := D_r$ {Sve što je ostalo u D_r je neispravno proglašeno nebitnim}

Odziv (engl. *recall*) pokazuje koliko smo bitnog teksta uzeli u odnosu na sav bitan tekst. Određuje se sa:

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

Mjera koja objedinjuje *točnost* i *odziv* je mjera *F1*:

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \tag{5.3}$$

Mjera $F1$ je poseban slučaj F_β mjere za $\beta = 1$. Ona daje jednak značaj preciznosti i odzivu.

$$F_\beta = \left(1 + \beta^2\right) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \tag{5.4}$$

5.1. Uzorci za evaluaciju

Za potrebe evaluacije ručno je očišćeno 49 HTML stranica. Kriteriji za preuzimanje stranica s Interneta nisu strogo određeni, ali su se uglavnom uzimale početne stranice portala, članci na portalu, blogu, raspravi na forumu ili stranice s uputama za korištenje nekog servisa ili programa. Popis stranica naveden je u dodatku B.

Odluka pripada li segment primarnom ili sekundarnom sadržaju data je na osnovu ljudske procjene. Nakon čišćenja, u HTML datotekama ostao je samo primarni sadržaj stranica.

Dodatno, pri čišćenju bilježeni su elementi i atributi za koje se pokazalo da su karakteristični opisnici primarnog, odnosno sekundarnog sadržaja. Pri izgradnji

tih popisa nisu definirana pravila proglašavanja opisnika karakterističnim, već se popis gradio prema okvirnoj procjeni često viđenih opisnika. Lista karakterističnih opisnika primarnog sadržaja navedena je u dodatku C, a sekunarnog sadržaja u dodatku D. Sličan postupak opisali su Hranj et al. (2009) te naveli širu listu opisnika primarnog i sekundarnog sadržaja.

5.2. Rezultati čišćenja

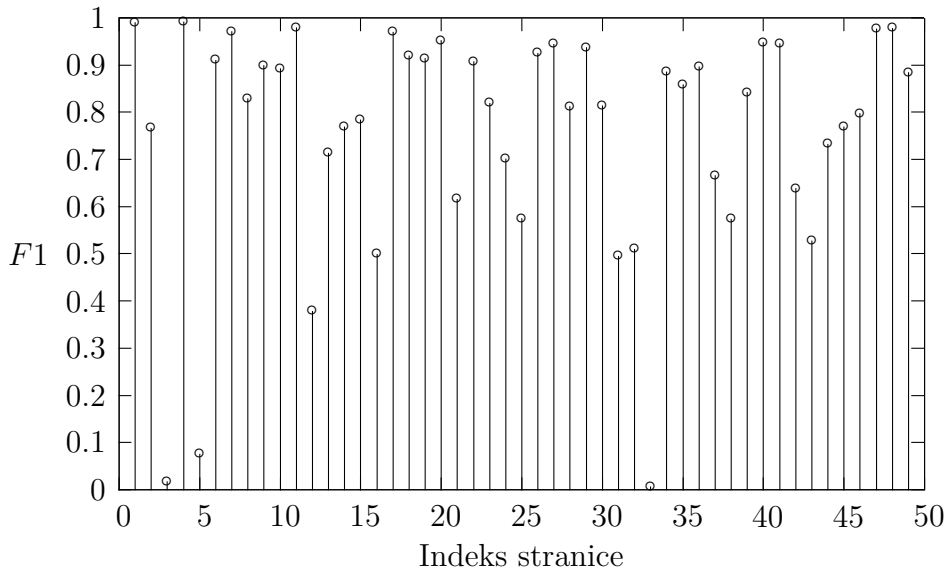
Algoritam je primjenjen nad test podatcima te su rezultati evaluirani metodom opisanom u 5. poglavlju.

Rezultati su prikazani grafički i tablično. U grafičkom prikazu x-os predstavlja indeks obrađene stranice koje se mogu vidjeti u dodatku B, a y-os uspješnost čišćenja izražena je mjerom $F1$.

5.2.1. Čišćenje po karakterističnim atributima

Grafički prikaz rezultata može se vidjeti na grafu 5.1.

Prosječna vrijednost $F1$ mjere je 0.760849. Detaljna statistika nalazi se u tablici A1.



Slika 5.1: Rezultati metode opisane u 4.1.

Diskusija

Rezultati su se pokazali relativno lošima. Najbolja karakteristika metode je razmjerno visoki odziv (89%).

Ekstremno loše očišćeni dokumenti su oni s indeksima 3, 5 i 33 (lista URL–ova navedena je u dodatku B).

Razlog su pojavljivanje nekih od karakterističnih atributa u vrhovnim elementima stranica. To su atributi koji su se pojavili kao opisnici primarnog sadržaja, a proglašeni su opisnicima za izbacivanje: `aduser-box`, `two-sidebars`. Ovakve se situacije možda mogu izbjeći uzimanjem u obzir dubine elmementa u DOM–u HTML dokumenta te omjera pozitivnih i negativnih karakterističnih atributa. Npr. ako je element *bolje* opisan pozitivnim atributima nego negativnim, onda se ne izbacuje. To dovodi do problema određivanja težina atributa i težina dubine elementa.

Budući da se metoda temelji na uklanjanju elemenata sekundarnog sadržaja radi povećanja preciznosti, teško je za očekivati kvalitetnije rezultate ako se pravila za odbacivanje postrože jer će to dovesti do značajnog smanjenja odziva.

Metoda može postati upotrebljiva ako joj se odziv maksimizira te se koristi u kombinaciji s drugim metodama.

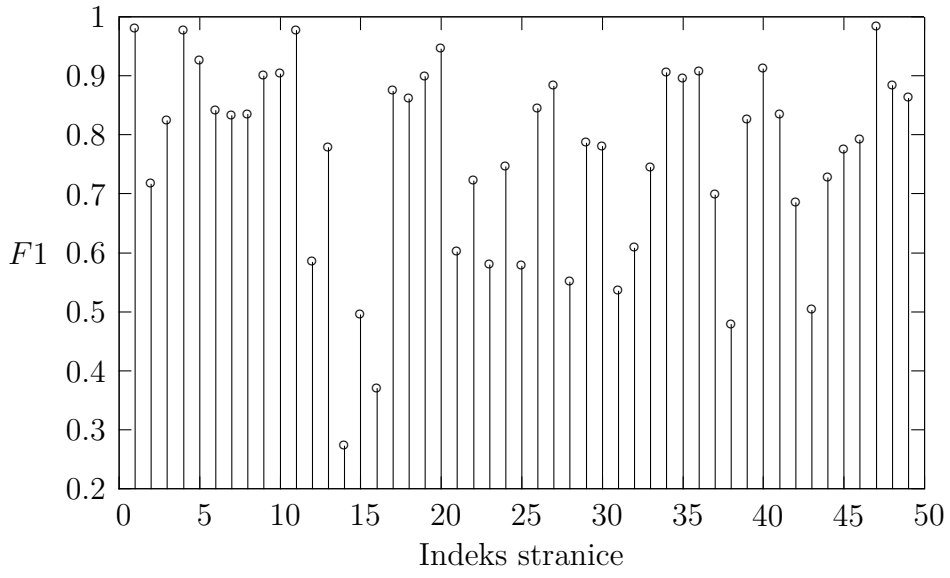
5.2.2. Čišćenje po gustoći elemenata

Grafički prikaz rezultata može se vidjeti na grafu 5.2. Prosječna vrijednost $F1$ mjere je 0.760849. Detaljna statistika nalazi se u tablici A2.

Diskusija

Metoda ima iznimno nisku preciznost (67%). Uz to, teško ju je kombinirati sa drugim metodama jer za potrebe određivanja primarnog sadržaja koristi potpuno DOM stablo stranice.

Da bi se popravio rezultat potrebno je definirati novi način određivanja gustoće dijelova stranice, odnosno promijeniti temelj izvedbe. Budući da je preciznost niska, a odziv izuzetno visok (96%!), potrebno je postrožiti pravila, odnosno naći nove koeficijente pri računanju trenutne gustoće.



Slika 5.2: Rezultati metode opisane u 4.2.

6. Implementacija

Kao jezik implementacije odabrana je Java.[1] Razlog odabira su Javina prenosivost i dostupnost velikog broja kvalitetnih open–source komponenti koje uvelike ubrzavaju razvoj aplikacije. Primjer gotovih komponenti je Jericho HTML parser[2] koji je, za razliku od većine drugih HTML parsera, otporan na neispravane HTML dokumente (koji su čest slučaj). Dostupan je pod *Eclipse Public License (EPL)*[3] i *GNU Lesser General Public License (LGPL)*[4] licencijama. Odnosno, komponentu je u komercijalnim aplikacijama dozvoljeno koristiti bez naknade ako način korištenja podliježe uvjetima navedenim u bar jednoj od licencija.

6.1. Komponente za čišćenje

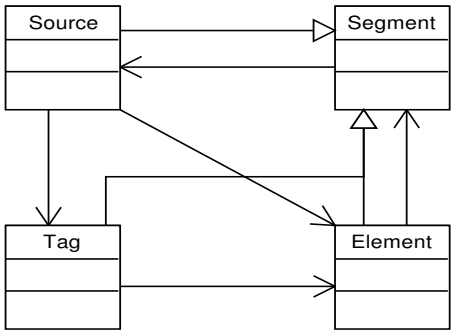
Implementacija metoda izvedena je uz pomoć parsera Jericho. Jericho nudi mogućnost dohvata HTML dokumenta preko `URL` razreda.

Parsanje se provodi kombinacijom jednostavnog pretraživanja teksta i učinkovitog prepoznavanja elemenata te pohranjivanja pozicija elemenata (većina ostalih parsera temelji se na stablima ili događajima; npr. *SAX*[5] i *DOM* parseri).

Dokument je predstavljen razredom `Source` koji se sastoji od objekata razreda `Segment`. Dijagramom 6.1 prikazan je odnos strukturnih elemenata dokumenta.

Do sadržaja se može doći metodama za dohvat strukturnih elemenata, npr. `getAllElements()` za potpune HTML elemente, `getAllTags()` za dijelove HTML elemenata koji služe kao opisnici ili `getSource()` za potpuni dokument. Navedene metode implementirane su u razredu `Segment`. Postoje inačice metode koje dohvaćaju samo jedan dio (npr. `getNextElement()`) i metode koje dohvaćaju posebne dijelove nekih dijelova, kao što je `getAttributes()` razreda `Element`.

[1] URL: http://java.sun.com/
[2] URL: http://jerichohtml.sourceforge.net/docs/index.html
[3] URL: http://www.eclipse.org/legal/epl-v10.html
[4] URL: http://www.gnu.org/copyleft/lesser.html
[5] URL: http://www.saxproject.org/event.html



Slika 6.1: Prikaz odnosa razreda koji predstavljaju dokument

Korisna tehnika jest iteriranje po segmentima pomoću razreda `NodeIterator`. Tehniku ilustrira sljedeći primjer:

```
Iterator<Segment> iter = segment.getNodeIterator();
while (iter.hasNext()) {
  Segment seg = iter.next();
  // Obrada segmenta
}
```

Za dohvat sadržaja postoje gotovi razredi kao što su razred `Renderer` i razred `TextExtractor`.

Razred `Renderer` pomoću svoje metode `toString()` može prikazati dokument na način kako ga prikazuju web preglednici.

Razred `TextExtractor` u svojoj metodi `toString()` odabir sadržaja koji ne treba prikazati radi na osnovu razultata metode `excludeElement()`. Korisnik Jericho HTML parsera tu metodu nadjačava u svojoj implementaciji razreda `TextExtractor` te na izuzetno jednostavan način koristi mogućnosti razreda `TextExtractor` uz novi način rada. Na primjer:

```
TextExtractor textExtractor=new TextExtractor(source) {
  public boolean excludeElement(StartTag startTag) {
    return   startTag.getName()==HTMLElementName.P
        || "control".equalsIgnoreCase(
        startTag.getAttributeValue("class"));
  }
};
```

6.1.1. Čišćenje po karakterističnim atributima

Čišćenje je provedeno iteriranjem po segmentima dokumenta, za što se koristio razred `NodeIterator`.

Za svaki segment vrši se provjera radi li se o objektu razreda `Tag` ili drugom tipu segmenta. Svaki segment koji nije tipa `Tag` ispisuje se pomoću razreda `Renderer`. Za svaki tag se pomoću Javinog razreda `Pattern` provjerava sadrži li atribute karakteristične za sekundarni sadržaj. Ukoliko je to istina, tag bi se preskočio a njime i svi elementi koje on sadrži (za to nam služi metoda `encloses()` razreda `Segment`).

```
Element el = tag.getElement();
String id = el.getAttributeValue("id");
String cls = el.getAttributeValue("class");

if ((cls != null && pattern.matcher(cls).matches())
    || (id != null && pattern.matcher(id).matches())) {
  drop = el;
  continue;
}
```

Svaki sljedeći segment koji zadovoljava svojstvo `drop.encloses(segment)` se preskače.

Regularni izraz koji se koristio za određivanje opisuje li atribut sekundarni sadržaj je kombinacija izraza: `.*login.*`, `.*member.*`, `.*nav.*`, `.*search.*`, `.*sidebar.*`, `.*info.*`, `.*foot.*`, `.*ftr.*`, `.*sponsor.*`, `.*buy.*`, `.*download.*`, `.*user.*`, `.*rss.*`, `.*copy.*`, `.*menu.*`.

Prije primjene izraza ulazni niz bi se prebacio u mala slova. Izrazi su kombinirani pomoću operatora *ili*.

6.1.2. Čišćenje po gustoći elemenata

Metoda za čišćenje je realizirana `TagDensityBasedExtractor` razredom. Obrada dokumenta provodi se kroz dva prolaza. Pri prvom prolazu stvori se stablasta struktura koja predstavlja stranicu i gustoću svakog pojedinog elementa. Čvorovi stabla su elementi privatnog razreda `Node`. Stablo se stvara iterativno pri obilasku DOM stabla stranice. Kod odgovoran za stvaranje stabla i pridjeljivanje gustoće čvorovima (elementima):

```
Node child = currentNode.addChild(seg, curScore);
    if (child.toString().length() > 1) {
        curScore -= Math.log10(child.toString().length());
    } else {
        curScore += 1.f;
    }
    currentNode = child;
} else {
    curScore = 0.f;
    while (!currentNode.encloses(seg)) {
        currentNode = currentNode.getParent();
    }
    Node child = currentNode.addChild(seg, curScore);
    currentNode = child;
}
```

Početna vrijednost trenutne gustoće je 0.0.

Ispis očišćene stranice dobiva se pozivom `treeToString()` metode nad korjenom novostvorene stablaste strukture. Očišćena stranica se ispisuje rekurzivnim obilaskom stabla sa gustoćama elemenata. Kod odgovoran za ispis stabla:

```
public String treeToString() {
    StringBuilder sb = new StringBuilder(seg.length());
    treeToString(this, sb);
    return sb.toString();
}

private void treeToString(Node root, StringBuilder sb) {
    if (root.getScore() < 0.f) {
        sb.append(root.toString());
    }

    for (Node node : root) {
        treeToString(node, sb);
    }
}
```

6.2. Komponenta za evaluaciju

Razvijena je modularna komponenta za evaluaciju. Temelj komponente čine apstraktni razredi `Evaluator` i `DataUnit`.

Implementiran je evaluator `LCSequenceEvaluator` koji kao mjeru usporedbe uzima duljinu najdužeg zajedničkog podniza. Vremenska složenost evaluacije je $O(nm)$, pri čemu su m i n duljine prvog, odnosno drugog dokumenta. Prostorna složenost je $O(\min(m, n))$ (asimptotski $\sim 2 \cdot \min(m, n)$). Traženje najdužeg zajedničkog podniza dano je algoritmom 4.

Algorithm 4 Traženje najdužeg zajedničkog podniza

Ulaz: a – sadržaj prvog dokumenta i b – sadržaj drugog dokumenta
Izlaz: Duljina najdužeg zajedničkog podniza.
if $length(a) > length(b)$ **then**
  $LCS(b, a)$
end if
$rowOld[length(a) + 1]$
$rowNew[length(a) + 1]$
for $(j := 1; j \leq length(b); inc(j))$ **do**
  $rowNew[0] := 0$
  for $(i := 1; i \leq length(a); inc(i))$ **do**
    if a[i-1] = b[j-1] **then**
      $rowNew[i] := rowOld[i - 1] + 1$
    else
      $rowNew[i] := \max(rowNew[i - 1], rowOld[i])$
    end if
  end for
  $swap(rowOld, rowNew)$
end for
return $rowOld[length(a)]$

Kao jedinica usporedbe mogu se odabrati riječ ili znak. Dokument razlučen na riječi predstavljen je razredom `StringData`, a dokument razlučen na znakove razredom `CharData`.

6.3. Corpus Collector – CoCo

Komponenta za čišćenje HTML dokumenata sastavni je dio aplikacije *Corpus Collector*. Corpus Collector, skraćeno *CoCo*, je aplikacija čija je svrha automatsko stvaranje korpusa iz dokumenata pobiranih sa web sjedišta. Detaljni opis se može naći u (Hranj et al., 2009).

7. Zaključak

U radu je opisana mogućnost korištenja weba kao izvora podataka za dubinsku analizu teksta. Opisane su metode koje se koriste pri čišćenju web stranica. Data je njihova usporedba te su uvedene kategorije: (1) metode koje rade nad web sjedištima (Yi et al., 2003; Bar-Yossef and Rajagopalan, 2002), (2) metode temeljne na vizualnim informacijama web stranica (Cai et al., 2003; Kovačević et al., 2002; Song et al., 2004) te (3) metode temeljene na strukturi pojedinačnih stranica (Arias et al., 2009; Finn et al., 2001). Predstavljena je, implementirana i evaluirana nova vrsta metode temeljena na karakterističnim atributima.

Opisani su česti problemi postojećih metoda poput loše obrade stranica čiji je sadržaj podjeljen u više blokova. Uvedena je podjela metoda po namjeni na (1) metode sa ciljem postizanja velike preciznosti radi izgradnje korpusa te (3) metode sa svrhom čišćenja stranica za krajnjeg korisnika (npr. uklanjanje reklama radi ugodnijeg korištenja ili prikaza stranica na uređajima s malim ekranima). Navedene su i metode koje ne čiste web stranice nego za svoje potrebe koriste rezultate web tražilica.

Opisan je problem evaluiranja metoda, date su mjere evaluacije i rezultati evaluiranja predstavljenih metoda. Radi bolje usporedbe s postojećim metodama, razvijene metode je potrebno evaluirati nad skupovima podataka korištenim pri razvoju drugih metoda.

U daljnjem radu metode je potrebno formalizirati te im poboljšati preciznost i odziv. Razviti nove metode i ispitati mogućnost kombinacije metoda. Pri razvoju novih metoda iskoristiti metode teorije informacija te ekstrakcije informacija.

Metode će se prilagoditi određenim potrebama, npr. izradi korpusa. Ispitati korištenje razvijenih metoda pri skupljanju mišljenja s weba (engl. *option mining*). Povezati razvijene metode u sustav za automatsko stvaranje korpusa iz dokumenata pobiranih sa web sjedišta—*Corpus collector*.

Literatura

Javier Arias, Moreno, Koen Deschacht, and Marie-Francine Moens. Language independent content extraction from web pages. In *Proceedings of the 9th Dutch–Belgian information retrieval workshop*. University of Twente, 2009.

Michele Banko and Eric Brill. Mitigating the paucity–of–data problem: Exploring the effect of training corpus size on classifier performance for natural language processing, 2001.

Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 580–591, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511522.

Marco Baroni and Motoko Ueyama. Building general–and special–purpose corpora by Web crawling. In *Proceedings of the 13th NIJL International Symposium, Language Corpora: Their Compilation and Application*, pages 31–40, 2006.

Gemma Boleda, Stefan Bott, Rodrigo Meza, Carlos Castillo, Toni Badia, and Vicente Lopez. Cucweb: a catalan corpus built from the web, 2006.

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. VIPS: a visionbased page segmentation algorithm. *Microsoft, Seattle, WA, Tech. Rep. No. MSRTR-2003-79*, 2003.

Aidan Finn, Nicholas Kushmerick, and Barry Smyth. Fact or fiction: Content classification for digital libraries, 2001.

Zoran Hranj, Ivan Karačić, Ivan Kmetović, Branimir Kocman, Ivan Krišto, Nikola Pleša, Josip Saratlija, and Nikola Šantić. *Corpus collector – tehnička dokumentacija*. Fakultet Elektrotehnike i Računarstva, 2009.

Adam Kilgarriff and Gregory Grefenstette. Web as corpus. In *Lancaster University*, pages 342–344, 2001.

Adam Kilgarriff and Gregory Grefenstette. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29:333–347, 2003.

Miloš Kovačević, Michelangelo Dilligenti, Marco Gori, and Veljko Milutinović. Recognition of common areas in a web page using visual information: a possible application in a page classification. In *in the proceedings of 2002 IEEE International Conference on Data Mining (ICDM'02*, page 250. IEEE Computer Society, 2002.

Nicholas Kushmerick. Learning to remove internet advertisements. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, pages 175–181, New York, NY, USA, 1999. ACM. ISBN 1-58113-066-X. doi: http://doi.acm.org/10.1145/301136.301186.

Bing Liu. Editorial: Special issue on web content mining. *ACM SIGKDD Explorations Newsletter*, 6:1–4, 2004.

Christopher Manning, D. and Hinrich Schtze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 1999. ISBN 0262133601.

Željko Panian. *Informatički enciklopedijski rječnik, @–L*, volume 1. Europapress holding, 2005.

Ruihua Song, Haifeng Liu, Wei-Ying Ma, and Wen Ji-Rong. Learning block importance models for web pages. In *In Intl. World Wide Web Conf. (WWW*, pages 203–211. ACM Press, 2004.

Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: http://doi.acm.org/10.1145/956750.956785.

Dodatak A
Statistika rezultata čišćenja

A1. Metoda opisana u poglavlju 4.1.

Tablica A1: Rezultati metode opisane u 4.1.

Indeks	Preciznost	Odziv	$F1$
1	0.99	0.99	0.99
2	0.63	0.99	0.77
3	0.97	0.01	0.02
4	1.00	0.99	0.99
5	0.87	0.04	0.08
6	0.88	0.95	0.91
7	0.95	1.00	0.97
8	0.89	0.78	0.83
9	0.83	0.99	0.90
10	0.81	0.99	0.89
11	0.97	1.00	0.98
12	0.37	0.40	0.38
13	0.85	0.61	0.71
14	0.63	0.99	0.77
15	0.80	0.77	0.79
16	0.34	0.94	0.50
17	0.94	1.00	0.97
18	0.86	1.00	0.92
19	0.86	0.97	0.92

(nastavak na sljedećoj stranici)

Tablica A1 – Nastavak

Indeks	Preciznost	Odziv	$F1$
20	0.94	0.97	0.95
21	0.45	0.99	0.62
22	0.84	1.00	0.91
23	0.70	0.98	0.82
24	0.54	1.00	0.70
25	0.42	0.93	0.58
26	0.88	0.98	0.93
27	0.91	0.99	0.95
28	0.69	1.00	0.81
29	0.89	1.00	0.94
30	0.69	0.99	0.81
31	0.33	0.99	0.50
32	0.34	0.99	0.51
33	0.94	0.00	0.01
34	0.80	0.99	0.89
35	0.98	0.77	0.86
36	0.82	0.99	0.90
37	0.50	0.99	0.67
38	0.41	0.99	0.57
39	0.73	1.00	0.84
40	0.92	0.98	0.95
41	0.98	0.92	0.95
42	0.48	0.95	0.64
43	0.38	0.85	0.53
44	0.58	0.99	0.73
45	0.63	0.99	0.77
46	0.67	0.99	0.80
47	0.96	1.00	0.98
48	0.97	0.99	0.98
49	0.84	0.94	0.89
Prosjek	0.75	0.89	0.76

A2. Metoda opisana u poglavlju 4.2.

Tablica A2: Rezultati metode opisane u 4.2.

Indeks	Preciznost	Odziv	$F1$
1	0.99	0.97	0.98
2	0.56	0.99	0.72
3	0.71	0.99	0.83
4	0.96	0.99	0.98
5	0.92	0.93	0.93
6	0.74	0.98	0.84
7	0.72	1.00	0.83
8	0.72	0.99	0.84
9	0.84	0.97	0.90
10	0.84	0.98	0.90
11	0.98	0.97	0.98
12	0.41	0.99	0.59
13	0.66	0.96	0.78
14	0.16	0.98	0.27
15	0.34	0.93	0.50
16	0.23	0.98	0.37
17	0.78	1.00	0.87
18	0.77	0.98	0.86
19	0.88	0.92	0.90
20	0.93	0.97	0.95
21	0.44	0.98	0.60
22	0.57	0.99	0.72
23	0.41	0.98	0.58
24	0.60	0.99	0.75
25	0.42	0.93	0.58
26	0.74	0.99	0.85
27	0.80	1.00	0.88

(nastavak na sljedećoj stranici)

Tablica A2 – Nastavak

Indeks	Preciznost	Odziv	$F1$
28	0.38	0.98	0.55
29	0.66	0.97	0.79
30	0.65	0.98	0.78
31	0.37	0.99	0.54
32	0.44	0.98	0.61
33	0.62	0.94	0.75
34	0.84	0.98	0.91
35	0.95	0.85	0.90
36	0.85	0.97	0.91
37	0.54	0.97	0.70
38	0.32	0.98	0.48
39	0.71	1.00	0.83
40	0.86	0.97	0.91
41	0.97	0.73	0.84
42	0.54	0.94	0.69
43	0.36	0.84	0.50
44	0.58	0.98	0.73
45	0.64	0.98	0.78
46	0.68	0.94	0.79
47	0.97	1.00	0.98
48	0.80	0.99	0.89
49	0.80	0.94	0.86
Prosjek	0.67	0.96	0.77

Dodatak B
Popis URL–ova stranica korištenih za evaluaciju

1. http://www2.roguewave.com/support/docs/hppdocs/stdug/11-3.html
2. http://www.nytimes.com/2008/12/12/business/12auto.html?bl&ex=1229144400&en=402e6793db93da56&ei=5087%0A
3. http://www.extremedreams.co.uk/index.php/Extremedreams/scad_diving/xsid/42
4. http://www.accelerateresults.com/category/2/article/158-six-smart-ways-to-grow-small-business-it?source=pcworld
5. http://www.pitchforkmedia.com/article/record_review/147938-the-welcome-wagon-welcome-to-the-welcome-wagon
6. http://www.post-gazette.com/pg/08331/930769-120.stm
7. http://tech.slashdot.org/tech/08/12/11/2046250.shtml
8. http://www.rollingstone.com/news/coverstory/24937978
9. http://www.touregypt.net/featurestories/anat.htm
10. http://www.jutarnji.hr/svijet/clanak/art-2008,12,11„144688.jl
11. http://behindthecloud.blog.hr/2007/11/index.html
12. http://www.billboard.com/bbcom/index.jsp
13. http://www.billboard.com/bbcom/reviews/index.jsp

14. http://www.roughguides.com/website/shop/products/British-Cult-Comedy.aspx

15. http://www.lonelyplanet.com/burkina-faso

16. http://www.rollingstone.com/reviews/album/24458350/review/24602349/call_and_response_the_remix_album

17. http://www.salon.com/opinion/paglia/2008/12/10/hillary_mumbai/

18. http://howto.wired.com/wiki/Cheat_on_the_Need_to_Sleep

19. http://www.greibachin_oblik.bloger.hr/

20. http://www.vjesnik.hr/html/2008/12/11/vijesti.asp#5

21. http://www.vjesnik.hr/html/2008/12/11/

22. http://www.developer.com/java/ent/article.php/3336761

23. http://www.popmatters.com/pm/column/65891-cut-to-the-whatever/

24. http://www.tportal.hr/funbox/funtime/2592/Devetogodisnjak-napisao-bestseler-o-zavodenju.html

25. http://www.nacional.hr/articles/view/50280/

26. http://ekologija.hr/index.php?id=166&tx_ttnews[tt_news]=973&tx_ttnews[backPid]=167&cHash=240297a666

27. http://ekologija.hr/index.php?id=166&tx_ttnews[tt_news]=971&tx_ttnews[backPid]=167&cHash=e926828efe

28. http://www.extremedreams.co.uk/

29. http://www.developer.com/java/other/article.php/1016841

30. http://forum.hr/showthread.php?t=38790

31. http://www.touregypt.net/godsofegypt/

32. http://news.google.com/

33. http://www.pitchforkmedia.com/

34. http://augustin.bloger.hr/

35. http://www.hr/wwwhr/abouthr/tradit/index.hr.html

36. http://www.monitor.hr/

37. http://www.extremeironing.com/modules.php?op=modload&name=Reviews&file=index&req=showcontent&id=12

38. http://www.pcworld.com/article/155362/its_official_google_chrome_exits_beta.html

39. http://www.slobodnadalmacija.hr/Split-%C5%BEupanija/tabid/76/articleType/ArticleView/articleId/33956/Default.aspx

40. http://www.vjesnik.hr/Info/Default.asp?r=kaz

41. http://en.wikipedia.org/wiki/List_of_Unicode_characters

42. http://www.zd-mioc.hr/forum/index.php?topic=3773.msg47093

43. http://www.nacional.hr/

44. http://www.net.hr/crnakronika/page/2008/12/11/0469006.html

45. http://www.net.hr/infocentar/zagreb/page/2008/12/02/0062006.html

46. http://en.wikipedia.org/wikplainnati/Netiquette

47. http://cortex.cs.nuim.ie/tools/spikeNNS/doc/node29.html

48. http://www.carnet.hr/sigurnost?CARNetweb=18680f1836bbe6791b37fd3ddab72599

49. http://www.ananova.com/news/story/sm_3117541.html

Dodatak C
Popis karakterističnih opisnika primarnog sadržaja

```
<div id="articleBody">,
<div id="article">,
<div class="content">,
<div id="authorId">,
<td class="navlist">,
<h1 class="header">,
<table class="storyPageColumns">,
<div class="storyBreadcrumbs">,
<div class="storyText">,
<div id="pagebody"
class="clear-block">,
<div id="mainbody"
class="clear-block">,
<div id="main">,
<div class="content description">,
<div id="pagecontent">,
<div class="story_body">,
<div class="commentBody">,
<div id="mainContent">,
<div id="contentBody">,
<div class="reviewcontent">,
<div class="ctl_body"
style="overflow:  hidden;">,
<div id="post_message_1254943">,
<div class="blogtext">,
<div class="blogtitle">,
<div class="blogdate">,
<p class="MsoNormal">,
<div class="NewsItem">,
```

Dodatak D
Popis karakterističnih opisnika sekundarnog sadržaja

, <ul id="memberTools">,
<ul class="tabs">, ,
, ,
,
,
<ul class="toolsList" id="toolsList">,
<a href="#secondParagraph"
class="jumpLink"> Skip to next paragraph,
,
,
,
,
,
,
,
,
<td class="navlist">,
<td class="infoBoxHeading",
<td><table class="infoBoxContents"
border="0" cellpadding="3" cellspacing="0"
width="100%">,
<td class="boxText">,
,
,
,
,
,
<ul class="buy">, <ul class="download">,
<div class="story_box_right",
,
, <ul id="user-utils">,
,
,
,
,
,
, ,
,
,
,
,
<li id="hiddens_26081681" class="hide">,
,
,
,
,
<table id="m_ctl5_hMenu1_2" class="menuHorizontal1"
style="width: 90px;" cellpadding="0" cellspacing="0"
height="28">,
<ul id="nav">,
,
<td class="Footer">,
,
,
,
,
,
,
,
,
,
<li class="rss">, ,
<A href="http:// www.itcareerplanet.com/"
class="menulink">,
,
<td class="tfoot">,
,
<ul id="navlist">,
,
,

Postupak čišćenja web stranica u svrhu dubinske analize teksta

Sažetak

Za razliku od tradicionalnih tekstovnih dokumenata, web stranice tipično sadržavaju veliku količinu informacija koje se ne odnose izravno na njihov sadržaj, poput promidžbenih poruka, navigacijskih uputa, i sl. U kontekstu dubinske analize teksta i računalno-lingvističke obrade, takve informacije predstavljaju neželjeni šum.

U okviru rada proučeni su postupci za automatsko čišćenje dokumenata u HTML-u od nepotrebnog sadržaja, razvijena programska implementacija postupka pogodna za ugradnju u pobirač dokumenata s web sjedišta te provodeno eksperimentalno vrednovanje postupka.

Ključne riječi: HTML, web stranice, uklanjanje šuma, automatsko čišćenje, dubinska analiza teksta

Web page cleaning techniques for text mining

Abstract

Unlike traditional text documents, web pages typically contain large amount of information that doesn't refer to content of web page directly, for example advertisements, navigation etc. In context of text mining and computational linguistic processing, such information represent unwanted noise.

This work describes automated web page cleaning techniques and presents program implementation and experimental evaluation of a cleaning technique suitable for using with web crawler.

Keywords: HTML, web pages, boilerplate removal, automated web cleaner, text mining