UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Barbara Tvrdi

Uporaba dejavnikov optimizacije spletnih strani za napovedovanje uvrstitev v iskalniku Google

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

University
of Ljubljana

Faculty of Computer
and Information Science

Tržaška 25
1000 Ljubljana, Slovenia
Phone: +386 1 476 84 11
Fax: +386 1 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01837/2012
Datum: 03.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BARBARA TVRDI**

Naslov: **UPORABA DEJAVNIKOV OPTIMIZACIJE SPLETNIH STRANI ZA NAPOVEDOVANJE UVRSTITEV V ISKALNIKU GOOGLE**

USE OF SEARCH ENGINE OPTIMIZATION FACTORS FOR GOOGLE PAGE RANK PREDICTION

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V diplomski nalogi preučite, ali je moč merljive karakteristike spletnih strani, za katere obstaja domneva, da vplivajo na rang teh v spletnih iskalnikih, uporabiti pri napovedovanju njihove uvrstitve v iskalniku Google. Pri raziskavi uporabite tehnike strojnega učenja. Hipoteza naloge je, da je na ta način možno identificirati ključne dejavnike, ki vplivajo na razvrstitev, in s tem poiskati tiste elemente spletnih strani, na katere bi se v namene izboljšanja uvrstitve razvijalci spletnih strani lahko osredotočili.

Mentor:

prof. dr. Blaž Zupan

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisana Barbara Tvrdi,

z vpisno številko 63060301,

sem avtorica diplomskega dela z naslovom:

Uporaba dejavnikov optimizacije spletnih strani za napovedovanje uvrstitev v iskalniku Google

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom
  prof. dr. Blaža Zupana
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 04.07.2012 Podpis avtorice:

Zahvala

Zahvalila bi se mentorju prof. dr. Blažu Zupanu za vse nasvete in pomoč pri izdelavi diplomskega dela. Za pomoč pri praktičnem delu se zahvaljujem tudi Lanu Žagarju.

Za neomejeno količino razumevanja, podpore in spodbude pri pisanju diplome pa bi se rada zahvalila svojemu zaročencu Janezu.

Na koncu se zahvaljujem še svojim staršem za vso podporo, ki so mi jo nudili že od prvih šolskih dni.

Kazalo

Seznam uporabljenih kratic in simbolov

SEO - Search Engine Optimization

URL - Uniform Resource Locator

HTML - HyperText Markup Language

IP - Internet Protocol

PR - PageRank

KNN - k-Nearest Neighbours

SVM - Support Vector Machine

AUC - Area Under Curve

ROC - Receiver Operating Characteristic

CA - Classification Accuracy

Povzetek

Spletni iskalniki so tekom let postali pomembno orodje za iskanje informacij. Znano je, da uporabniki v kar 62% primerov izberejo povezavo na prvi strani rezultatov iskanja. Postopki optimizacije spletnih strani omogočajo izboljšavo spletne strani in posledično boljšo uvrstitev v iskalniku. Natančne specifikacije dejavnikov, ki vplivajo na uvrstitev, izdelovalci spletnih iskalnikov ne želijo razkriti. V tem diplomskem delu smo zato poskusili izbrati nekaj, v literaturi najbolj pogosto omenjenih, dejavnikov za optimizacijo v iskalniku Google. Z njihovo pomočjo smo z uporabo metod strojnega učenja poskusili zgraditi model, ki bi lahko napovedal, ali bi se neka spletna stran uvrstila med prvih 10 rezultatov iskanja (tj. na prvo stran iskalnika). Najboljše rezultate je dosegla metoda uvrščanja v skupine z naključnimi gozdovi, vendar je bila dobljena ocena AUC pod mejo sprejemljive ocene AUC za tovrstne probleme. Poskusili smo poiskati še statistično značilne informativne značilke, a je bilo teh v izbrani množici malo, imele pa so tudi zelo nizko informativnost. Za boljše rezultate bi bilo potrebno uporabiti še kakšne druge značilke in povečati množico učnih primerov.

Ključne besede:

spletni iskalniki, optimizacija spletnih strani, strojno učenje

Abstract

Over the years, search engines have become an important tool for finding information. It is known that users select the link on the first page of search results in 62% of the cases. Search engine optimization techniques enable website improvement and therefore a better ranking in search engines. The exact specification of the factors that affect website ranking is not disclosed by search engine owners. In this thesis we tried to choose some most frequently mentioned search engine optimization factors for Google search engine. Using the factors we tried to apply machine learning methods to build a model that predicts whether a site would be ranked among the top 10 search results (i.e. the first page of search engine results). The best results were achieved using a classification method called random forests, but the obtained AUC was below acceptable AUC estimates for such problems. We also tried to find statistically significant informative features. Only a few features matched the criteria, but had a very low information content. To achieve better results other features could be used and the number of training examples could be increased.

Key words:

search engines, search engine optimization, machine learning

Poglavje 1

Uvod

Spletni iskalniki so s svojo rastočo vlogo v človekovem življenju postali pomembno orodje za iskanje informacij ter učinkovito sredstvo za trženje. Raziskovalna hiša IDC je ocenila, da kar 84% uporabnikov interneta dobi informacije, ki jih iščejo, prek spletnih iskalnikov [5]. Zato morajo biti spletni iskalniki narejeni tako, da uporabniku prikažejo čim bolj relevantne rezultate. Poleg tega ponujajo oglasni prostor za plačane oglase, vendar med uporabniki še vedno prevladuje izbira organskih rezultatov.

Podjetja, ki želijo, da jih spletni iskalnik uvrsti čim višje na seznamu organskih rezultatov iskanja, vlagajo denar v optimizacijo spletnih strani (*angl. Search Engine Optimization - SEO*). Raziskava iProspect and Jupiter Research je pokazala, da 62% uporabnikov izbere povezavo s prve strani rezultatov, 90% uporabnikov pa s prvih treh strani rezultatov [2]. Ker spletni iskalniki posodabljajo in izpopolnjujejo svoje algoritme, je poznavanje dejavnikov SEO pomembno za ohranjanje dobre uvrstitve med rezultati iskanja.

Podrobnosti delovanja algoritmov lastniki spletnih iskalnikov ne razkrivajo, saj bi s tem prišlo do razvrednotenja kvalitete iskalnih rezultatov in potencialnih prevar. V literaturi in na spletu obstajajo le mnenja strokovnjakov s področja SEO, kateri dejavniki pomembno vplivajo na uvrstitev med rezultati iskanja. Nekateri izmed teh podatkov o dejavnikih so mogoče že zastareli, nekateri podatki so zgolj domneve posameznikov, nekaj pa je takih, za katere strokovnjaki na podlagi preteklih izkušenj sklepajo, da vplivajo na uvrstitev.

Po podatkih podjetja NetMarketshare je imel v aprilu 2012 največji delež iskanj na globalni ravni iskalnik Google, in sicer kar 81% [7]. Zato so v di-

plomskem delu zbrani in opisani najbolj pogosto omenjeni dejavniki v zvezi s tem iskalnikom. S pomočjo metod strojnega učenja smo nato poskusili zgraditi učni model, ki bi lahko napovedal, ali bi se neka spletna stran uvrstila med prvih 10 rezultatov iskanja, torej na prvo stran rezultatov iskalnika. Če bi tak model obstajal, bi lahko ugotovili, kateri dejavniki odločilno vplivajo na boljšo uvrstitev v iskalniku.

Diplomsko delo obsega uvodno poglavje, tri osrednja poglavja in zaključek. V drugem poglavju je na kratko predstavljeno delovanje spletnih iskalnikov. Naslednje poglavje obsega seznam in podroben opis dejavnikov SEO, ki so bili izbrani za analizo. Četrto poglavje opisuje način zajema testnih podatkov in merjenja dejavnikov ter uporabljene metode strojnega učenja. Na koncu poglavja so priloženi rezultati testov. V zaključku naloge so povzeti in kritično ocenjeni bistveni rezultati analize, ki jim sledi še nekaj idej za morebitne izboljšave.

Poglavje 2

Delovanje spletnih iskalnikov

Spletni iskalniki morajo hraniti ogromno količino informacij. Te morajo biti uporabniku dostopne v čim krajšem času in čim bolj ustrezati kriterijem iskanja. Zaradi lažjega obvladovanja in vzdrževanja vseh funkcionalnosti so spletni iskalniki razdeljeni na tri bistvene naloge:

- iskanje in zbiranje vsebine, odkrite na spletu s pomočjo spletnih pajkov (*angl. web crawlers*),
- indeksiranje in shranjevanje vsebine v iskalniku bolj prijazni obliki,
- iskanje ustreznih rezultatov za iskalni niz [1].

2.1 Iskanje s pomočjo spletnih pajkov

Spletni pajek (ali krajše pajek) je program, ki raziskuje splet in išče ter vrača vsebine za indeksiranje in shranjevanje. Spletni iskalniki imajo sezname že znanih obstoječih povezav URL, ki jih pajki obiskujejo eno za drugo. Preden vsebino predajo naprej v indeksiranje, preiščejo stran, da najdejo izhodne povezave. Če pajek najde nove povezave, jih doda v globalni seznam povezav, ki jih morajo pajki obiskati.

Da bi zagotovili čim večjo količino obiskanih strani, pajek vedno preveri, kdaj je bila stran nazadnje obiskana. Če je bila indeksirana pred kratkim, potem jo preskoči in da prednost drugim stranem. Spletni iskalnik Google pa ima tudi posebnega pajka, imenovanega FreshBot, ki obiskuje strani, kjer se vsebina pogosto spreminja.

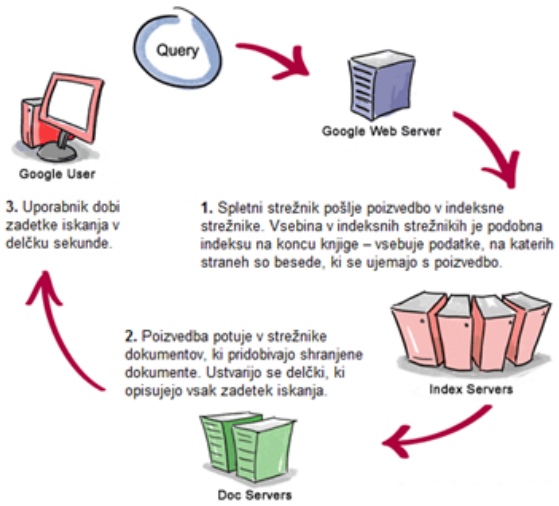
Strani, ki nimajo zunanjih povezav, niso nikoli obiskane, saj pajki ne najdejo poti (povezav) do njih. Takim stranem se reče nevidni splet (*angl. invisible web*). Da bi pajek našel stran, mora do nje voditi vsaj ena zunanja povezava s strani, ki jo pajki obiščejo [1].

2.2 Indeksiranje vsebine

Proces indeksiranja skrbi za izdelavo/osveževanje kataloga besed in njihovih pozicij na vsaki strani, ki jo pajek odkrije. Katalogi omogočajo zelo hitro iskanje po spletu. Edina slabost takega pristopa je, da iskanje poteka po nekoliko starejših podatkih in ne po tistih, ki so trenutno v resnici na voljo [1].

2.3 Iskanje rezultatov

Ko uporabnik v spletni iskalnik vnese iskalni niz, ga spletni strežnik pošlje indeksnim strežnikom, ki imajo shranjene kataloge. Iskalnik najprej iz kataloga izbere strani, ki so relevantne (tj. se njihova vsebina do neke mere ujema z iskalnim nizom) in jih razvrsti po pomembnosti. Pomembnost je določena z oceno zaupanja in številom zunanjih povezav strani [2]. Izbrane strani posredujejo strežnikom, ki hranijo vsebine strani. Iz teh vsebin iskalnik naredi izrezke, ki jih spletni iskalnik v določenem vrstnem redu prikaže uporabniku kot rezultat iskanja [1]. Potek procesiranja iskalne zahteve prikazuje slika 2.1.



Slika 2.1: Potek procesiranja iskalne zahteve uporabnika do prikaza rezultatov

Poglavje 3

Dejavniki optimizacije spletnih strani

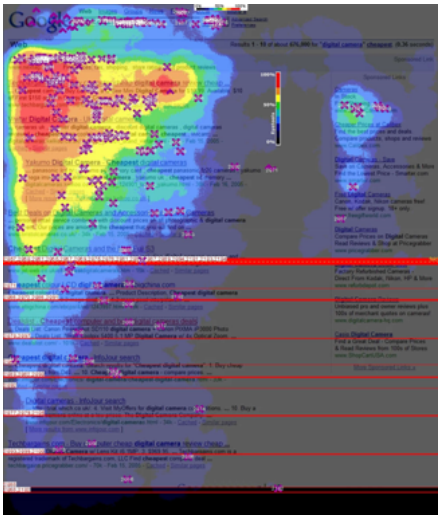
3.1 O optimizaciji spletnih strani

Po definiciji je optimizacija spletnih strani proces, s katerim povečamo število obiskovalcev spletne strani s tem, da poskrbimo, da se stran pojavlja višje na seznamu rezultatov iskanja v spletnem iskalniku. Višje kot se spletna stran pojavlja v seznamu rezultatov, večja je verjetnost, da jo bo uporabnik obiskal [5]. Med preostale, za to raziskavo sicer nepomembne naloge SEO pa spada še privabljanje pravih obiskovalcev, nudenje prave vsebine, zadrževanje obiskovalcev na strani, ... [4]

Zakaj je tako pomembno, da je spletna stran čim višje na seznamu zadetkov, nam kažeta raziskava iProspect and Jupiter Research, ki je bila omenjena že v uvodu naloge, in Eye Tracking Research, ki so jo opravila podjetja Enquino, Eyetools in Didit. V prvi raziskavi so ugotovili, da 62% uporabnikov izbere povezavo s prve strani rezultatov, 90% uporabnikov pa ne gre nikoli dlje od 3. strani rezultatov. Poleg tega so ugotovili, da 36% uporabnikov misli, da so podjetja, ki zasedajo prva mesta na seznamu rezultatov, vodilna v panogi. Pri drugi raziskavi so ugotavljali, kakšna je fiksacija pogleda na seznamu rezultatov pri Googlu. Iz slike 3.1 lahko vidimo, da uporabniki največ časa gledajo levi zgornji del strani. Prav tako se dobro vidi, da organski rezultati dobijo veliko več pozornosti kot plačani oglasi [2].

Prvi korak pri SEO je izdelava seznama ključnih besed, za katere želimo spletno stran optimizirati. Pod pojmom ključna beseda razumemo besedo ali besedno

zvezo, ki jo vpišemo v spletni iskalnik, ko hočemo najti določeno informacijo. Izbiri ključnih besed sledi optimizacija strani za posamezno ključno besedo, kar pomeni, da preverimo dejavnike SEO in stran ustrezno popravimo (optimiziramo). Dejavniki SEO se delijo v dve veliki skupini, in sicer na dejavnike optimizacije na spletni strani (*angl. on-site SEO factors*) in na dejavnike optimizacije izven spletne strani (*angl. off-site SEO factors*) [5].



Slika 3.1: Fiksacija pogledov uporabnikov na strani rezultatov

3.2 Dejavniki optimizacije na spletni strani

Optimizacija na spletni strani se navezuje na optimizacijo elementov, ki so sestavni del spletne strani in jih lahko lastnik samostojno spreminja (naslovi, besedila, metaoznake, slike, ...) [5]. V nadaljevanju sledi seznam in kratek opis dejavnikov, ki se najbolj pogosto pojavljajo v literaturi.

Ključna beseda v naslovu strani
Naslov strani določimo v znački <TITLE>v glavi dokumenta HTML. Naslov strani se pojavi na vrhu okna brskalnika in na seznamu rezultatov iskanja, zato je pomemben tudi za privabljanje obiskovalcev. Če je ključna beseda v naslovu strani, obstaja večja verjetnost, da jo bo Google uvrstil višje na seznamu rezultatov. V nekaterih virih razlikujejo dva dejavnika, in sicer glede na to, ali je ključna beseda prva beseda v naslovu strani ali pa se pojavi kjerkoli v naslovu strani.

Ponavljajoči se naslovi strani
Če ima veliko naših strani enak ali skoraj enak naslov, se to obravnava enako kot podvojena vsebina in ima negativen vpliv na uvrstitev strani v iskalniku.

Število vseh besed v naslovu strani
Po testiranju, ki je opisano v [6], imajo najbolje uvrščene strani v povprečju 7.8 besed v naslovu strani. Google na svojem seznamu prikaže prvih 64 znakov naslova, zato daljši naslovi niso priporočljivi.

Ključna beseda v meta opisu strani
Opis strani se prav tako kot naslov prikaže na seznamu rezultatov iskanja. Če ne napišemo opisa strani, večina spletnih iskalnikov uporabi kar prvih 30-40 besed na strani, kar lahko odvrne potencialne obiskovalce. Google prikaže opis le, če je v njem najdena ključna beseda, sicer sam zgenerira kratek povzetek iz besedila, kjer je našel ključno besedo.

Število besed v meta opisu strani
Po testiranju, ki je opisano v [6], imajo najbolje uvrščene strani v povprečju 16 besed v meta opisu strani. Po priporočilih naj bi opis strani vseboval 165 znakov ali manj.

Ključna beseda v meta ključnih besedah
Včasih je bil ta dejavnik najbolj pomemben za določanje relevance spletne strani, danes pa ga Google v celoti ignorira in rezultate iskanja ocenjuje glede na vsebino strani, zunanje povezave in druge dejavnike. Nekateri drugi spletni iskalniki (npr. Yahoo! in MSN) pa še vedno upoštevajo tudi meta ključne besede, zato razvijalci za mnoge spletne strani še vedno izpolnijo to značko.

Ključna beseda v vsebini strani
Število ponovitev ključne besede v vsebini strani ne sme biti preveliko, saj iskalniki ta pojav obravnavajo kot pretirano rabo ključnih besed (*angl. keyword stuffing*), zaradi česar je lahko stran kaznovana in uvrščena nižje na seznamu rezultatov. Gostoto ključnih besed na strani izračunamo tako, da število ponovitev ključne besede delimo s številom vseh besed na strani (tu so izključene besede iz naslova strani, meta opisa in ključnih besed, komentarjev HTML, značk alt, ...). Priporočljiva gostota ključnih besed se giblje med 3-6%.

Ključna beseda v znački H1
Značke H (najpogosteje se uporabljajo H1, H2 in H3) se uporablja za označevanje naslovov in podnaslovov v vsebini strani. Spletni iskalniki jih smatrajo kot pomembno informacijo o tematiki strani. V virih razlikujejo dva dejavnika pri znački <H1>, in sicer glede na to, ali se ključna beseda pojavi na začetku značke ali pa kjerkoli v znački <H1>.

Ključna beseda v ostalih značkah H (H2 – H6)
Ta dejavnik je manj pomemben kot dejavnik za značko <H1>, vendar najdena ključna beseda v drugih naslovnih značkah vseeno lahko pripomore k boljši uvrstitvi na seznamu rezultatov.

Ključna beseda v atributu alt značke <IMG>
Značko <IMG>uporabljamo za vstavljanje slik v dokument HTML. V atribut alt naj bi vnesli kratek opis vsebine slike. To besedilo se uporabniku prikaže samo v primeru, ko brskalnik slike ne more prikazati. Atribut alt so včasih izkoriščali tudi za kopičenje ključnih besed, zato je danes njegov pomen močno upadel.

Ključna beseda v imenu slike
Če stran vsebuje slike, ki so poimenovane z nekimi opisnimi, pomenljivimi imeni, lahko na ta način izboljšamo svojo uvrstitev v rezultatih iskanja. Iskalniki namreč ne morejo vedeti, kaj slika predstavlja, lahko pa si pomagajo z imenom datoteke, ki vsebuje sliko.

Ključna beseda v značkah <B>in <STRONG>
Ker z značkama <B>in <STRONG>poudarimo neko besedo ali besedno zvezo (uporabniku se namreč prikaže krepko obarvan tekst), jih tudi spletni iskalniki uporabljajo kot znak, da je poudarjena beseda bolj pomembna.

Ključna beseda v značkah <I>in <EM>
Znački <I>in <EM>uporabljamo za poševno besedilo, s čimer podobno kot pri znački <B>poudarimo določene besede ali besedne zveze.

Ključna beseda v značkah <LI>
Značko <LI>uporabljamo, ko želimo narediti seznam elementov. Ta del besedila je prav tako poudarjen, oziroma izpostavljen in zato ga iskalniki obravnavajo podobno kot značke <B>in <I>.

Ključna beseda v besedilu notranje povezave
Če je na strani povezava do druge strani v sklopu iste spletne strani, je pomembno, da za besedilo povezave izberemo dober opis, ki zajema vsebino strani, na katero vodi povezava. Iskalniki besedila povezav upoštevajo kot precej pomembne dejavnike.

Ključna beseda v besedilu izhodne povezave
Če stran vsebuje povezavo do neke druge spletne strani, je prav tako pomembno, da izberemo dobro besedilo povezave.

Ključna beseda v imenu domene
Če bo spletni iskalnik našel ključno besedo v imenu domene (npr. kbeseda.si), potem bo ta stran zelo visoko uvrščena na seznamu zadetkov.

Ključna beseda v imenu poddomene
Ta dejavnik je podoben zgoraj omenjenemu, le da imajo poddomene manjšo pomembnost kot domene. Primer: kbeseda.domena.si.

Ključna beseda v nazivu strani v URL
Z uporabo prepisovanja naslovov URL (*angl. URL rewriting*) lahko ustvarimo lahko berljive naslove URL, kjer za mape in nazive strani uporabimo izraze, ki opisujejo vsebino strani (npr. domena.si/kbeseda.html).

Ključna beseda v nazivu mape v URL
Podobno kot zgoraj lahko za nazive map uporabimo izraze, ki pomensko določajo njihovo vsebino (npr. domena.si/kbeseda/stran.html).

Uporaba notranjih okvirjev
Spletni iskalniki ignorirajo vsebino notranjih okvirjev (*angl. inside frames*), saj smatrajo, da je ta vsebina iz drugih strani in zato ne more biti unikatna vsebina strani, ki jo indeksirajo. Če imamo v notranjih okvirjih svojo vsebino, je iskalnik ne bo zaznal in naše strani ne bo prikazal na seznamu rezultatov.

Koda HTML po standardih W3C
Čeprav iskalniki ne upoštevajo neposredno tega dejavnika, je veljavna koda HTML nujno potrebna za zagotavljanje pravilne interpretacije elementov spletne strani za potrebe indeksiranja spletnega iskalnika.

Število pokvarjenih povezav
Povezave do drugih strani morajo biti pravilne, zato je potrebno občasno preveriti, ali vse povezave na strani vodijo do obstoječih strani.

Nekateri strokovnjaki za optimizacijo spletnih strani trdijo, da imajo dejavniki na spletni strani 40% vpliv na končno uvrstitev strani na seznamu rezultatov [5]. Večji vpliv naj bi imeli dejavniki optimizacije izven spletne strani. Med njimi je kot najpomembnejši omenjen PageRank, ki bo bolj podrobno opisan v naslednjem poglavju. Kljub temu pa je dobro skrbeti za optimizacijo na spletni strani, saj se lahko tudi na ta način ustvari prednost pred konkurenco, še posebej, če konkurenčne strani niso dobro optimizirane [5, 2, 1, 8].

3.3 Dejavniki optimizacije izven spletne strani

Pri optimizaciji izven spletne strani se vsi procesi izvajajo izven strani in nad njimi pogosto nimamo popolnega nadzora. Predvsem sta pri tovrstni optimizaciji pomembna število in pomembnost zunanjih povezav, ki jih ima stran [5].

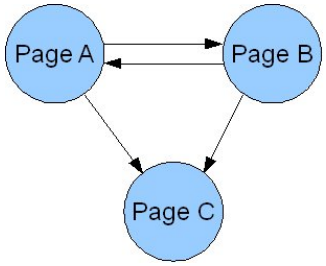
Ključna beseda v besedilu zunanje povezave
Pomembno je, s katerimi besedami je opisana povezava, ki vodi do naše strani. Če dobro povzame vsebino strani, je lahko stran bolje uvrščena na seznamu zadetkov.

PageRank
Mnogi uvrščajo PageRank med najpomembnejše dejavnike, ki vplivajo na uvrstitev strani. PageRank (PR) je algoritem, ki so ga razvili pri podjetju Google in upošteva tako število zunanjih povezav kot tudi njihovo pomembnost.

Svetovni splet lahko ponazorimo z usmerjenim grafom (slika 3.2). Vsaka spletna stran je v tem grafu predstavljena z vozliščem. Usmerjene povezave med vozlišči pa ponazarjajo povezave (*angl. links*) do drugih spletnih strani (vozlišč). Enačba (3.1) prikazuje osnovno formulo za izračun PR, ki sta jo Brin in Page predstavila v [16], pri čemer so $T_1 ... T_n$ strani, ki imajo povezavo do strani A, $C(T_i)$ število vseh izhodnih povezav na strani T_i in d faktor dušenja. Faktor dušenja d je verjetnost, da bo uporabnik sledil verigi povezav na straneh in ne bo odšel na naključno stran. Največkrat vzamemo *d=0.85*.

$$PR(A) = (1-d) + d\sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)} \tag{3.1}$$

S pomočjo grafa na sliki 3.2 lahko bolj nazorno razložimo enačbo (3.1). Stran C ima dve vhodni povezavi, ki štejeta kot glasova. Vsak glas je utežen s količnikom PR in števila vseh povezav na strani, ki daje glas. Tako so več vredni glasovi s strani z dobrim PR, ki nimajo veliko izhodnih povezav, sicer se vrednost PR razdrobi. Utežene glasove seštejemo in jih množimo s faktorjem dušenja.



Slika 3.2: Primer usmerjenega grafa

Enačba (3.2) prikazuje drugo verzijo algoritma za izračun PR. Vidimo lahko, da je edina razlika glede na enačbo (3.1) v tem, da faktor (1-d) delimo z N, ki predstavlja število vseh strani svetovnega spleta. Faktor $\frac{(1-d)}{N}$ sedaj res predstavlja verjetnost, da uporabnik pride naključno na to stran (ne prek povezave, ki vodi do te strani). Po tem izračunu vse vrednosti PR tvorijo verjetnostno porazdelitev po spletnih straneh, vsota vseh vrednosti PR pa je 1 [16, 9].

$$PR(A) = \frac{(1-d)}{N} + d\sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)} \tag{3.2}$$

Google dobljene vrednosti PR pretvori v lestvico vrednosti med 0 in 10. Obstajajo ugibanja, da ta lestvica ni linerna, temveč logaritemska, torej da je vsako stopnjo težje doseči. Vrednost 10 je najboljša in jo doseže le malo strani. PR 4 ali več se smatra kot dobra ocena, dočim nižje vrednosti pomenijo slabo ocenjeno stran.

PageRank ni odvisen od ključne besede trenutnega iskalnega niza, saj Google PageRank izračuna in shrani v posebnih izračunih, ki jih opravlja neodvisno od izvedenih iskanj.

Starost domene
Spletni iskalniki upoštevajo tudi starost domene, na kateri je spletna stran. Starejša kot je domena, več možnosti ima stran, da bo uvrščena višje v seznamu rezultatov.

Namenski naslov IP
Po nekaterih testiranjih v [6] naj bi imele vse visoko uvrščene strani namenske naslove IP (*angl. dedicated IP*). Namenski IP je dodeljen izključno eni domeni.

Sitemap
Ta dejavnik prav tako ni neposredno upoštevan, vendar lahko posredno vpliva na število indeksiranih strani, saj spletni pajek obišče vse strani, ki jih najde v tej datoteki na strežniku [6, 8].

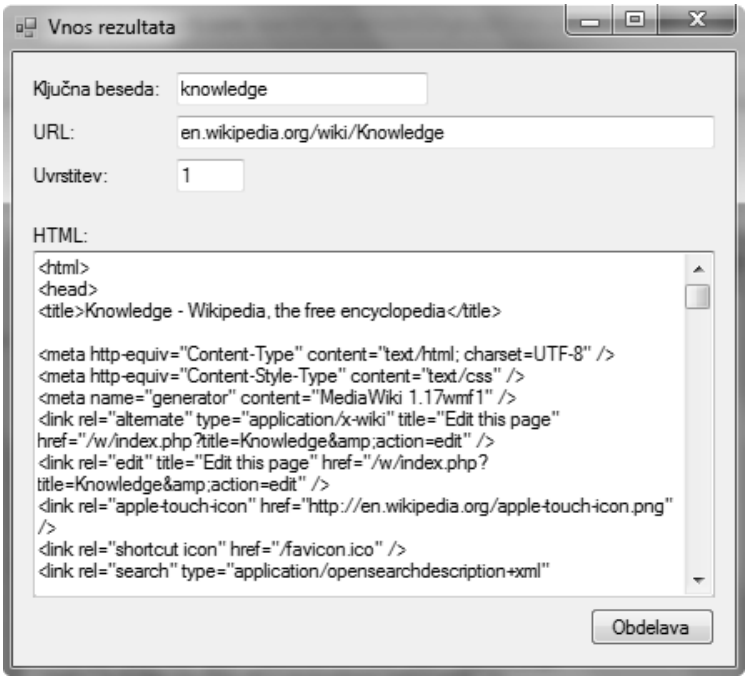
Poglavje 4

Uporaba dejavnikov SEO za napovedovanje uvrstitev

4.1 Izbor podatkov

Prvi korak pri zajemu testnih podatkov je bil izbor ključnih besed. Te smo izbrali popolnoma naključno. Zaradi poenostavitve merjenja dejavnikov smo se odločili za ključne besede, sestavljene iz ene angleške besede. Tako smo se izognili obravnavanju sklonov besed (če bi izbrali slovensko besedo) in vrstnega reda (če bi izbrali več besed). Končni izbor je sestavljalo 17 ključnih besed, in sicer: *knowledge, warehouse, union, ecology, photography, sharing, neuron, decoding, computer, planning, security, extinction, phenomena, tourist, sale, community* in *archery*.

Naslednji korak je bil vnos vsake izmed ključnih besed v iskalnik Google in merjenje dejavnikov za prvih 60 rezultatov iskanja. Za potrebe pridobivanja in obdelave podatkov smo v programskem okolju Microsoft Visual Studio 2010 razvili program, ki bi celoten proces opravil avtomatsko. Vendar se je izkazalo, da je tak postopek nemogoče izvesti, saj Google blokira vse avtomatizirane zahteve proti strežniku in s tem onemogoči dostop do rezultatov. Tako je bilo potrebno vsako ključno besedo ročno vnesti v iskalnik Google in potem prav tako ročno vnesti podatke o vsakem izmed prvih 60 rezultatov v naš program, ki smo ga dopolnili z vnosno masko. Primer izpolnjene vnosne maske prikazuje slika 4.1. Vnesti je bilo potrebno ključno besedo, za katero smo dobili ta rezultat, URL od rezultata, uvrstitev (na katerem mestu od 1 do 60 je bil ta rezultat) ter HTML strani. Na tem mestu je potrebno opomniti, da smo vnesli HTML posnetka (*angl. cache*) strani, če je bil ta na voljo, sicer



Slika 4.1: Maska za vnos podatkov o posameznem rezultata iskanja

smo vzeli HTML originalne strani. Posnetek strani, ki je na voljo ob rezultatu iskanja kot to prikazuje slika 4.2, smo vzeli zato, ker je tako izgledala stran, ko jo je Google nazadnje ocenjeval, kar je med drugim vplivalo na trenutno uvrstitev. Teoretično bi se namreč lahko stran vmes že spremenila in bi upoštevali HTML, ki ga Google še ni obravnaval.

Slika 4.2: Izsek rezultata iskanja z označeno povezavo do posnetka strani

Zadnji del obdelave podatkov je lahko ostal avtomatski. Program je namreč opravil merjenje dejavnikov na osnovi podatkov, podanih na vnosni maski. Podroben opis merjenja posameznih dejavnikov podaja naslednje podpoglavje.

Vse meritve so bile shranjene v eno datoteko, ki je razmejena s tabulatorji in primerna kot vhodna datoteka za orodje Orange [15]. Datoteka z meritvami

je po koncu meritev vsebovala 1020 primerov (1 primer vsebuje meritve za en rezultat iskalnika).

4.2 Merjenje dejavnikov

Za potrebe merjenja dejavnikov, ki izhajajo iz vsebine spletne strani, tj. kode HTML, ki smo jo shranili za vsak rezultat iskanja, smo s pomočjo knjižnice SgmlReaderDll [13] kodo HTML pretvorili v objekt tipa XmlDocument. Pri tem objektu lahko dobro izkoristimo funkcionalnost XPath, ki nam olajša iskanje značk in njihove vsebine. Primer, kako preprosto najdemo značko <TITLE>v dokumentu XML, je prikazan v kodi spodaj.

```
var tagNode = node.SelectSingleNode("//title");
```

Nekatere dejavnike lahko združimo v skupine glede na način njihovega merjenja.

V največjo skupino spadajo dejavniki, kjer je postopek merjenja sledeč:

- s pomočjo sintakse XPath poiščemo značke HTML, za katere merimo dejavnik,
- v njihovi vsebini ali vsebini atributa značk poskušamo najti obravnavano ključno besedo,
- če ena izmed značk vsebuje ključno besedo, potem ima ta dejavnik vrednost "Y", sicer ima vrednost "N".

Na tak način je program meril vrednosti naslednjih parametrov: *ključna beseda v naslovu strani*, *ključna beseda na začetku naslova strani*, *ključna beseda v meta opisu strani*, *ključna beseda v znački H1*, *ključna beseda na začetku značke H1*, *ključna beseda v ostalih značkah H*, *ključna beseda v atributu alt značke <IMG>*, *ključna beseda v imenu slike*, *ključna beseda v značkah <B>in <STRONG>*, *ključna beseda v značkah <I>in <EM>*, *ključna beseda v značkah <LI>*, *ključna beseda v besedilu notranje povezave* in *ključna beseda v besedilu izhodne povezave*.

Naslednja skupina dejavnikov uporablja preprosto štetje besed - vsebino obravnavane značke razbijemo na besede in preštejemo točne ponovitve iskane ključne besede. Končna vrednost dejavnika je številka. Taki dejavniki so *število besed*

v naslovu strani, *število besed v meta opisu strani* in *število ključnih besed v vsebini strani*. Da bi pravilno prešteli število ključnih besed v vsebini strani, je bilo potrebno iz kode HTML najprej odstraniti značke in pustiti samo njihovo vsebino.

Dejavniki iz naslednjega sklopa so povezani z obravnavo povezave URL do strani. Program iz naslova URL izlušči naziv poddomene (če ta obstaja), domene, imena map in naziv strani. Če vzamemo za primer URL:
http://www.test.domena.si/mapa1/mapa2/str.html,
bo program najprej poiskal domeno (domena), nato poddomeno (test), vse mape vrnil v seznamu (mapa1,mapa2) in našel naziv strani (str). Nato bo za vsak dejavnik pogledal ali je ključna beseda v ustreznem delu naslova URL in vrnil vrednost "Y", če je, sicer bo vrnil vrednost "N". Na tak način so bili izmerjeni naslednji dejavniki: *ključna beseda v imenu domene*, *ključna beseda v imenu poddomene*, *ključna beseda v nazivu strani v URL* in *ključna beseda v nazivu mape v URL*.

Za preostale dejavnike pa ni enotnega načina merjenja, zato bodo opisani vsak posebej.

Dejavnik *uporaba notranjih okvirjev* predstavlja število značk <IFRAME>, ki se nahajajo na strani.

Število napak po standardih W3C dobimo tako, da pošljemo zahtevo na naslov http://validator.w3.org/check?uri=X in namesto X navedemo URL strani. Validator v odgovoru vrne celotno specifikacijo napak v obliki kode HTML, iz katere s pomočjo XPath lahko hitro poiščemo mesto, kjer je navedeno skupno število napak.

Število pokvarjenih povezav ugotovimo tako, da poiščemo vse povezave na strani in preverimo, kakšno povratno kodo vrne strežnik po poslani zahtevi na stran, kamor kaže povezava. Če dobimo povratno kodo "Bad request"ali "Forbidden", potem je povezava pokvarjena.

Za ugotovaljanje *starosti domene* program pošlje zahtevo na naslov
http://www.whois.net/whois/X, kjer X predstavlja domeno. Iz odgovora, ki ga pošlje strežnik, nato izlušči datum, ko je bila domena ustvarjena. Končna vrednost dejavnika je število mesecev od datuma, ko je bila domena ustvarjena, do datuma obdelave.

Ali ima domena *namenski naslov IP* lahko preverimo tako, da pošljemo zahtevo na naslov http://www.ip-adress.com/reverse_ip/X, kjer namesto X navedemo ime domene. Iz odgovora lahko izluščimo število domen, ki so dodeljene temu naslovu IP. To število je tudi končna vrednost dejavnika.

Da bi ugotovili, kolikokrat se *ključna beseda* pojavi *v zunanji povezavi*, pošljemo zahtevo na google.com, kjer v iskalno okence vpišemo "link:X", kjer X predstavlja URL strani, ki je trenutno v obdelavi. Program obišče prvih 60 rezultatov iskanja in za vsakega poskusi v vsebini strani poiskati povezavo do obdelovane strani. Če najde tako povezavo, preveri, ali je v opisu povezave tudi obravnavana ključna beseda. Vrednost dejavnika je število zunanjih povezav, kjer je v opisu povezave uporabljena ključna beseda.

PageRank dobimo tako kot je opisano na [12]. Vrednosti dejavnika so lahko števila od 0 do 10.

Dejavnikov *ponavljajoči se naslovi strani* in *sitemap* ni bilo mogoče programsko pridobiti, zato sta izpuščena iz meritev.

Končna testna datoteka je obsegala 1020 primerov, 27 značilk (dejavnikov) in diskretno razredno spremenljivko. Razredna spremeljivka je vsebovala uvrstitev med rezultati iskanja in je lahko zasedala vrednosti od 1 do 60.

4.3 Postopki testiranja

Cilj vseh testov je bil najti tak model, ki bi lahko s sprejemljivo točnostjo napovedoval uvrstitve v iskalniku. Poskusili smo dva idejno različna pristopa - učenje rangiranja (*angl. learning to rank*) in uvrščanje v skupine (klasifikacijo). Podroben opis obeh pristopov se nahaja v podpoglavjih, ki sledita. Ocenjevanje značilk smo opravili zato, da bi morebitne slabe značilke izločili in s tem izboljšali učni model.

4.3.1 Učenje rangiranja

Cilj algoritmov za učenje razvrščanja je avtomatsko generiranje modela za rangiranje iz učnih podatkov. Model nato iz testnih podatkov zgradi permutacije uvrstitev primerov, ki naj bi bile čim bolj podobne pravim uvrstitvam [10].

Opisani postopek je primeren tudi za uporabo na učnih podatkih, ki smo jih pridobili iz iskalnika Google. Če bi lahko algoritem iz teh učnih podatkov zgradil dovolj dober model, bi ta lahko napovedoval uvrstitve posameznih testnih (novih) primerov. V ta namen smo uporabili algoritem SVM^{rank} [14].

Testni datoteki smo dodali še en atribut, ki označuje posamezne sklope rezultatov. Prvih 60 rezultatov je bilo povezanih s prvo ključno besedo, zato imajo enako oznako tega atributa itd. S tem omogočimo najboljši izkoristek funcionalnosti algoritma SVM^{rank}, ki zna delati s sklopi podatkov. Hkrati pa je tudi bolj smiselno, da se algoritem uči na sklopu uvrstitev, ki so med seboj povezane, in da ne meša med seboj uvrstitev za različne ključne besede. Ko imamo podatke tako razdeljene, lahko uporabimo funkcijo iz paketa Orange test_with_indices, ki za določene vhodne podatke za vsak sklop zgradi učni model iz podatkov preostalih sklopov in testira na trenutnem sklopu.

Za vrednotenje rezultatov smo poskusili izračunati koeficient utežene korelacije uvrstitev, ki izhaja iz koeficienta Spearmanove korelacije uvrstitev [11].

$$r_w = 1 - \frac{6 \sum_{i=1}^{n} (R_i - Q_i)^2 ((n - R_i + 1) + (n - q_i + 1))}{n^4 + n^3 - n^2 - n} \tag{4.1}$$

Spearmanov koeficient nam pomaga ugotoviti korelacijo med dejansko (pravilno) razvrstitvijo rezultatov in razvrstitvijo, ki jo vrne učni model. Utežili smo ga zato, ker je pri konkretnem primeru uvrstitev v iskalniku zelo pomembno, da so prvi primeri pravilno razvrščeni, manj pa je pomembno, če se učni model zmoti pri napovedi kasnejših uvrstitev. Pri izračunu smo uporabili formulo (4.1), implementacijo pa prikazuje spodnja koda.

```
from statc import rankdata,sumdiffsquared

def sumwdiffsquared(x,y):
    sds = 0
    n = len(x)
    for i in range(n):
        w = 2*n+2-x[i]-y[i]
        sds = sds + (x[i]-y[i])**2 * w
    return sds

def wspear(x,y):
    if len(x) <> len(y):
        raise ValueError, 'Input values not paired.'
    n = len(x)
```

```
    rankx = rankdata(x)
    ranky = rankdata(y)
    dsq = sumwdiffsquared(rankx,ranky)
    rs = 1 - 6*dsq / float(n*(n**3+n**2-n-1))
    return rs
```

4.3.2 Klasifikacija

Metode strojnega učenja uporabljamo za gradnjo klasifikatorjev iz učnih podatkov. Učni primeri so opisani z množico značilk (neodvisne zvezne ali diskretne spremenljivke) in razredom (odvisna diskretna spremenljivka). Naloga klasifikatorja je, da za nek nov primer določi, kateremu izmed možnih razredov pripada [3].

Za testiranje s klasifikacijskimi algoritmi smo uporabili enako vhodno datoteko kot za učenje razvrščanja. Tudi za klasifikacijo smo razdelili testne primere v sklope glede na ključno besedo, za katero so bili pridobljeni. Poskusili smo zmanjšati število vrednosti, ki jih zasede razredna spremenljivka. V originalnih podatkih je razredna spremenljivka lahko zasedala vrednosti od 1 do 60. Naredili smo 4 verzije vhodne datoteke, in sicer tako, da smo mejo za delitev postavili pri prvih 5, 10, 15 in 20 rezultatih. Če je bil primer uvrščen znotraj te meje, je dobil vrednost 1, sicer je zasedel vrednost 0. Na tak način smo želeli ugotoviti, ali lahko dobimo model, ki bi zanesljivo napovedal, ali bi se nek primer uvrstil med prvih 5, 10, 15 ali 20 rezultatov iskanja.

Postopek testiranja prikazuje spodnja koda. Primere iz vhodne datoteke smo razdelili na 17 sklopov. Nato smo izbrali en sklop, ki je bil namenjen testiranju. Iz preostalih sklopov smo zgradili 16 učnih modelov in na njih testirali primere iz izbranega sklopa. Končno napoved razreda za testni primer smo dobili tako, da smo vzeli najbolj pogosto napoved klasifikatorjev. Preizkusili smo nekaj najpogostejših metod strojnega učenja - večinski klasifikator, naivni Bayes-ov klasifikator, odločitvena drevesa, klasifikator z najbližjimi sosedi, klasifikator po metodi podpornih vektorjev in naključne gozdove. V nadaljevanju so na kratko predstavljeni posamezni algoritmi.

```
def make_query_indices(data, qid):
    # split data into 'folds' by query qid
    q2ind = dict((q, i) for i, q in enumerate(set(ex[qid].value
        for ex in data)))
    return [q2ind[ex[qid].value] for ex in data]
```

```
def dbmean(list):
    if len(list) == 0:
        return float('nan')
    sumP1 = 0
    sumP2 = 0
    for el in list:
        if el != '␣':
            sumP1 = sumP1 + el[0]
            sumP2 = sumP2 + el[1]
    return [float(sumP1)/len(list),float(sumP2)/len(list)]

def score(res, f=lambda x, y: wspear(x, y)):
    # average score of function f on experiment results
    scoress = [[] for i in range(res.numberOfLearners)]
    for r in Orange.evaluation.scoring.split_by_iterations(res)
       :
        actual = [i+1 for i in range(60)]
        for i, scores in enumerate(scoress):
            tmp = f(actual, [pr[0] for pr in [te.probabilities[
               i] for te in r.results]])
            scores.append(tmp)
    return [float(sum(scores)) / len(scores) for scores in
       scoress]

def testLearners(file):
    # data
    data = Orange.data.Table(file)
    qid = 'query'

    # split data by keywords
    inds = make_query_indices(data, qid)

    # get all classifiers
    clsNames = ["major","bayes","tree","svm","knn","forest"]
    cls = [[],[],[],[],[],[]]
    for fold in range(17):
        learnset = data.select(inds, fold)
        cls[0].append(Orange.classification.majority.
           MajorityLearner(learnset))
        cls[1].append(Orange.classification.bayes.NaiveLearner(
           learnset))
        cls[2].append(Orange.classification.tree.TreeLearner(
           learnset))
        cls[3].append(Orange.classification.svm.SVMLearner(
           learnset))
        cls[4].append(Orange.classification.knn.kNNLearner(
```

```
        learnset, k=10))
    cls[5].append(Orange.ensemble.forest.
        RandomForestLearner(learnset, trees=400))

# merge classifiers' predictions to one
mode = lambda x: max([(x.count(y),y) for y in x])[1]

# build ExperimentResults
expResults = Orange.evaluation.testing.ExperimentResults(
    iterations=17, classifierNames=clsNames, classValues
    =[0,1], weights=False)
clres = []
for fold in range(17):
    testset = data.select(inds, fold)
    for ex in testset:
        tex = Orange.evaluation.testing.TestedExample(
            iterationNumber=fold, actualClass=int(ex.
            getclass()))
        clspred = [[],[],[],[],[],[]]
        clsprob= [[],[],[],[],[],[]]
        for i in range(17):
            if i!=fold:
                for c in range(len(cls)):
                    tres = cls[c][i](ex, Orange.
                        classification.Classifier.GetBoth)
                    clspred[c].append(tres[0])
                    clsprob[c].append(tres[1])

        # add results to TestedExample
        for c in range(len(cls)):
          tex.add_result(mode(clspred[c]), dbmean(clsprob[c
              ]))
        clres.append(tex)

# add final results
expResults.results = clres

# calculate CA and AUC
cas = Orange.evaluation.scoring.CA(expResults)
aucs = Orange.evaluation.scoring.AUC(expResults)
scores = score(expResults, lambda x, y: wspear(x, y))

# print results
print 'learner\t\tCA\tAUC\tCC'
print '---------------------------------------'
for learner, ca, auc, cc in zip(expResults.classifierNames,
    cas, aucs, scores):
```

```
        print '%s:\t\t%.4f\t%.4f\t%.4f' % (learner, ca, auc, cc
            )
```

Večinski klasifikator (*angl. majority classifier)*
Algoritem v fazi učenja izračuna distribucijo vrednosti razredne spremenljivke in si jo zapomni. Vse testne primere uvrsti v večinski razred učnih podatkov. Pogosto se ta klasifikator uporablja za primerjavo z drugimi, naprednejšimi učnimi modeli. Če ima drug klasifikator boljšo točnost kot večinski klasifikator, to pomeni, da imamo model, ki zna nekaj več kot samo preprosto uvrščati primere v večinski razred.

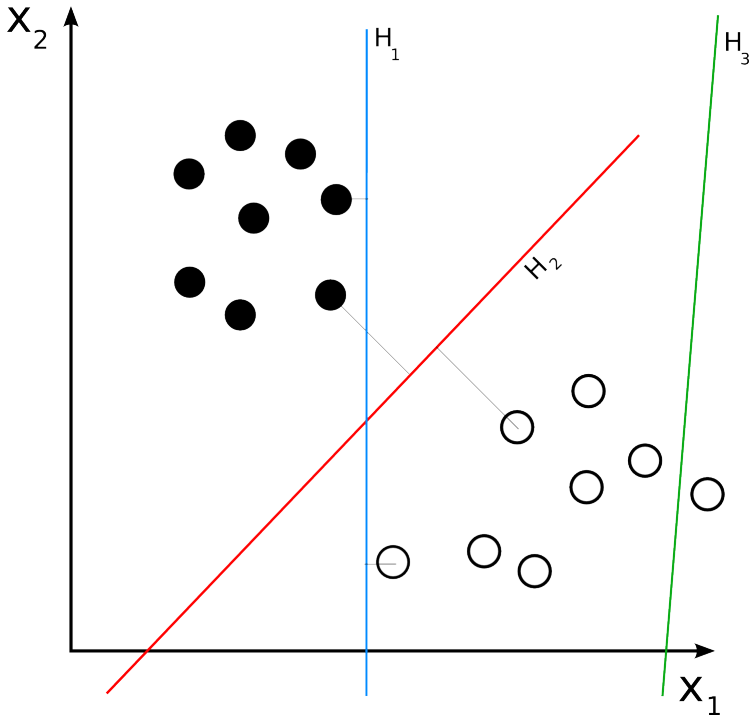
Naivni Bayesov klasifikator (*angl. naive Bayesian classifier)*
Bayesov klasifikator je enostaven verjetnostni klasifikator, ki za izračune uporablja Bayesov teorem. Naziv "naivni"ima zaradi predpostavke, da so značilke med seboj neodvisne. Naloga klasifikatorja je, da za vsak nov testni primer izračuna pogojne verjetnosti za vse možne razrede glede na učne podatke in vrednosti značilk testnega primera [3].

Odločitvena drevesa (*angl. decision trees)*
Algoritem za gradnjo odločitvenega drevesa izbira značilke glede na njihovo pomembnost in tako gradi odločitvena pravila. V vsakem koraku izbere značilko, ki glede na izbrano mero (informacijski prispevek, reliefF, gini indeks, ...) najbolje razdeli učne primere v dve skupini. Skupek odločitvenih pravil lahko predstavimo v obliki drevesa, kjer korensko vozlišče predstavlja najbolj pomemben atribut, listi drevesa pa predstavljajo razrede [3].

Klasifikator z najbližjimi sosedi (*angl. nearest neighbours classifier)*
KNN (klasifikator z najbližjimi sosedi) spada med lene algoritme. V fazi učenja namreč ne zgradi modela, temveč si samo shrani vse učne primere. Ko dobi testni primer za klasifikacijo, poišče k najbližjih sosedov (učnih primerov) in pogleda njihove razrede. Testnemu primeru določi prevladujoči razred sosedov [3].

Klasifikator po metodi podpornih vektorjev (*angl. Support Vector Machine)*
Pri metodi SVM (metoda podpornih vektorjev) so učni primeri predstavljeni kot točke v prostoru. V fazi učenja mora algoritem najti tako hiperravnino, ki optimalno razdeli razreda. Učni primeri iz vsakega razreda, ki so najbližje tej hiperravnini, se imenujejo podporni vektorji. Algoritem poskuša maksimizirati razdaljo med podpornimi vektorji in s tem minimizirati možnost napake

pri klasifikaciji novih primerov. Učni model nato tudi testni primer preslika v ta prostor točk in mu glede na to, na katero stran hiperravnine pade, določi razred [3].



Slika 4.3: Določanje optimalne hiperravnine za razdelitev učnih primerov

Na sliki 4.3 vidimo množici učnih primerov iz dveh različnih razredov. Hiperravnina H_3 je slaba, saj ne loči pravilno učnih primerov v dva razreda. H_1 in H_2 obe dobro ločita primere, vendar bi bila v tem primeru H_2 boljša, saj maksimizira razdaljo med mejnimi učnimi primeri.

Naključni gozdovi (*angl. random forests)*
Algoritem zgradi več odločitvenih dreves z omejeno izbiro najboljše značilke iz naključno izbrane podmnožice značilk. Za nek testni primer glasujejo vsa drevesa iz gozda, končni razred pa predstavlja razred, za katerega je glasovalo največ dreves [3].

Kvaliteto vseh zgoraj omenjenih klasifikatorjev smo izrazili s klasifikacijsko točnostjo in AUC (area under curve). Klasifikacijska točnost je delež pravilnih klasifikacij testnih primerov v primerjavi s številom vseh testnih primerov.

AUC predstavlja površino pod ROC (receiver operating characteristic) krivuljo. Večja kot je vrednost AUC, bolj kvaliteten je klasifikator. ROC krivulja prikazuje razmerje med občutljivostjo (relativna frekvenca pravilno klasificiranih pozitivnih primerov) in specifičnostjo (relativna frekvenca pravilno klasificiranih negativnih primerov). AUC je verjetnost, da bo klasifikator med naključno izbranim pozitivnim in negativnim primerom, višje uvrstil pozitivni primer [3].

4.3.3 Ocenjevanje značilk

S postopkom ocenjavanja značilk smo želeli izločiti tiste značilke, ki so nekoristne. Na tak način bi lahko izboljšali učni model, ker bi upošteval samo kvalitetne značilke.

Za merjenje kvalitete značilk smo uporabili mero ReliefF. Največja prednost tega algoritma pred drugimi je, da zajame tudi informacijo iz med seboj odvisnih značilk, poleg tega pa zna obravnavati tudi šumne in neznane vrednosti in večrazredne probleme. Algoritem za vsak primer iz naključne podmnožice učnih primerov poišče k najbližjih primerov iz vsakega možnega razreda. Nato posodobi kvaliteto vsake značilke glede na to, ali loči primere iz istega razreda (nezaželjena lastnost) in ali loči primere iz različnih razredov (zaželjena lastnost) [3].

Vrednosti, ki jih vrne ReliefF kot ocene kvalitete značilk, imenujemo statistike. Postavili smo ničelno hipotezo H_0, ki se glasi:
Vrednost te statistike smo pridobili na naključnih podatkih.
Če lahko dokažemo, da za neko značilko statistike niso prišle iz naključnih podatkov, potem je taka značilka dobra, oziroma kaže na neko odvisnost v podatkih.

Za ocenjevanje značilk smo uporabili permutacijski test. Postopek je prikazan v spodnjem algoritmu. Najprej za vsako značilko iz vhodnih podatkov izračunamo originalno vrednost mere ReliefF. Potem 1000-krat naključno premešamo vrednosti značilk v podatkih in vsakič izračunamo nove vrednosti ReliefF in si jih shranimo. Na koncu imamo za vsako značilko poleg originalne vrednosti ReliefF še 1000 vrednosti, ki smo jih dobili z naključnim mešanjem podatkov. Iz teh podatkov izračunamo vrednost p, ki predstavlja delež naključnih ocen, ki so večje od originalne vrednosti ReliefF. Nato postavimo še mejo α, ki določa, do katere vrednosti p lahko ničelno hipotezo zavrnemo in

označimo atribut kot pomemben.

```
import Orange, orange
import random

def ShuffleAttributes(file):
    data = Orange.data.Table(file)
    for i in range(len(data.domain.attributes)):
        attvals = [ex[i] for ex in data]
        random.shuffle(attvals)
        for ex,attval in zip(data, attvals):
            ex[i] = attval
    return data

def TestAttributes(file):
    nshuf = 1000

    # load data
    data = Orange.data.Table(file)

    # measure original ReliefF
    meas = orange.MeasureAttribute_relief(k=10, m=20)
    origRelief = [meas(attr, data) for attr in data.domain.
       attributes]

    # shuffle attributes and measure ReliefF
    shufRelief = [[] for i in range(len(data.domain.attributes)
       )]
    for i in range(nshuf):
        data_s = ShuffleAttributes(file)
        meas = orange.MeasureAttribute_relief(k=10, m=20)
        for a in range(len(shufRelief)):
            shufRelief[a].append(meas(data_s.domain.attributes[
               a], data_s))

    # calculate p
    ps = []
    for i in range(len(origRelief)):
        nL = 0.0
        for j in range(nshuf):
            if (shufRelief[i][j] > origRelief[i]):
                nL = nL + 1
        print nL
        ps.append(nL/nshuf)

    # output
    print 'attribute\t\tReliefF\tp'
    print '-----------------------------------------------'
```

```
    for attr, reliefF, p in zip(data.domain.attributes,
        origRelief, ps):
         print '%s:\t\t%.4f\t%f' % (attr.name, reliefF, p)
```

4.4 Rezultati

Prvi test smo naredili z algoritmom za učenje rangiranja - SVM^{rank}. Poskusili smo uporabiti dve različni vrednosti za parameter c, ki predstavlja kompromis med napako pri učenju in razdaljo med podpornimi vektorji (*angl. margin*) [14]. Rezultati so prikazani v tabeli 4.1.

klasifikator	koeficient utežene korelacije
SVM^{rank} (c = 0.1)	0.2206
SVM^{rank} (c = 1.0)	**0.2243**

Tabela 4.1: Koeficienti utežene korelacije za algoritem SVM^{rank}

Vidimo lahko, da je koeficient utežene korelacije približno 0.22, kar je zelo slabo. Tak klasifikator ne napoveduje dovolj natančno vrstnega reda uvrstitev.

Pri uporabi algoritmov za klasifikacijo smo naredili več različnih testov. Najprej smo preverili razliko v kvaliteti klasifikatorjev, če poteka učenje na 16 sklopih učnih primerov hkrati in testiranje na enem sklopu (16x60) ter na vsakem od 16 sklopov ločeno, kjer potem iz vsakega sklopa dobimo klasifikator, ki glasuje za klasifikacijo primerov iz testnega sklopa (16x1x60). Za testiranje smo uporabili vhodno datoteko z mejo delitev razredne spremenljivke pri vrednosti 10. To mejo smo večkrat uporabili pri testih, saj je pomensko zanimiva. Predstavlja namreč odgovor na vprašanje, ali lahko klasifikator dovolj uspešno napove, ali se bo neka stran uvrstila med prvih 10 zadetkov na iskalniku - na prvo stran iskalnika. Pri algoritmu kNN smo parametru k dodelili vrednost 10, pri naključnih gozdovih (forest) pa smo uporabili 400 dreves. Dobljene klasifikacijske točnosti (CA) in AUC prikazuje tabela 4.2. S tem testom smo želeli preveriti, ali je učenje bolj učinkovito, če se klasifikator uči iz sklopa podatkov, ki so posledica uporabe iste ključne besede v iskalniku.
Pristop z učenjem iz posameznih sklopov se je pokazal kot boljši, zato smo ga uporabili pri naslednjem testu. Želeli smo primerjati kvaliteto klasifikatorjev glede na to, pri kateri vrednosti postavimo mejo za delitev razredne spremenljivke. Rezultati so prikazani v tabeli 4.3.

klasifikator		16x60	16x1x60
majority	CA	0.8333	0.8333
	AUC	0.5000	0.5000
bayes	CA	0.7137	0.7941
	AUC	0.6197	0.6364
tree	CA	0.8324	0.8343
	AUC	0.5011	0.5625
SVM	CA	0.8333	0.8333
	AUC	0.5000	0.6249
kNN	CA	0.8029	0.8333
	AUC	0.5517	0.6205
forest	CA	0.8402	0.8343
	AUC	0.6478	0.6543

Tabela 4.2: Primerjava dveh pristopov h klasifikaciji testnih primerov

klasifikator	uvrstitev <= 5		uvrstitev <= 10		uvrstitev <= 15		uvrstitev <= 20	
	CA	AUC	CA	AUC	CA	AUC	CA	AUC
majority	0.9167	0.5000	0.8333	0.5000	0.7500	0.5000	0.6667	0.5000
bayes	0.9127	0.6758	0.7941	0.6197	0.6618	0.6123	0.5912	0.5944
tree	0.9176	0.6453	0.8343	0.5625	0.7510	0.5627	0.6676	0.5669
SVM	0.9167	0.5000	0.8333	0.6249	0.4333	0.5000	0.6667	0.5502
kNN	0.9167	0.6514	0.8333	0.6205	0.7627	0.6274	0.6765	0.5921
forest	0.9167	**0.6973**	0.8343	**0.6543**	0.7539	0.6349	0.6765	0.6115

Tabela 4.3: Primerjava kvalitete klasifikatorjev glede na mejo za razred

Po pregledu rezultatov lahko ugotovimo, da kvaliteta klasifikatorjev pada z večanjem meje za delitev razredne spremenljivke. Najboljše rezultate dobimo pri meji 5 za naključne gozdove, saj je tam AUC 0.6973, kar je ravno na meji 0.7, ki je spodnja meja sprejemljivega klasifikatorja. Tudi pri meji 10 se najbolje obnesejo naključni gozdovi, vendar je vrednost AUC 0.6543, kar je pod mejo sprejemljivega.

Na koncu smo želeli preveriti še razliko v kvaliteti klasifikatorja za učenje rangiranja (SVM^{rank}) in ostalimi klasifikatorji. Da bi lahko rezultate primerjali, smo morali uporabiti enake mere. Algoritem za učenje rangiranja smo spremenili tako, da smo prvih 10 uvrščenih primerov označili z razredom 1, ostale pa z razredom 0. To nam je omogočilo enostaven izračun klasifikacijske

točnosti in AUC. Pri rezultatih smo navedli samo vrednosti, ki smo jih dobili za *c=1.0*. Pri ostalih algoritmih smo uporabili vhodno datoteko z mejo za delitev razreda 10 in nad dobljenimi rezultati izračunali koeficient utežene korelacije (CC). Rezultate lahko vidimo v tabeli 4.4.

klasifikator	CA	AUC	CC
majority	0.8333	0.5000	**0.5000**
bayes	0.7373	0.6150	0.1940
tree	0.8304	0.5646	0.0938
SVM	0.8333	0.6204	0.2095
kNN	0.8333	0.6205	0.1797
forest	0.8343	**0.6543**	**0.2212**
SVM^{rank}	0.7745	**0.6345**	**0.2243**

Tabela 4.4: Primerjava kvalitete SVM^{rank} glede na druge klasifikatorje

Ob primerjavi klasifikatorja SVM^{rank} z ostalimi klasifikatorji vidimo, da je SVM^{rank} po korelacijskem koeficientu boljši od drugih klasifikatorjev, razen od večinskega klasifikatorja. Šele ko bi našli tak klasifikator, ki bi imel boljši korelacijski koeficient od večinskega klasifikatorja, bi lahko rekli, da imamo dober klasifikator, saj zna primere uvrščati glede na neko znanje in ne samo po preprostem principu večinskega razreda. Če upoštevamo mero AUC, pa se SVM^{rank} obnese slabše kot naključni gozdovi.

Zadnji test je bil ocenjevanje značilk. Za testiranje smo uporabili vhodno datoteko z mejo za delitev razreda pri 10. Tabela 4.5 prikazuje rezultate za vsako značilko - vrednost mere ReliefF in vrednost p. Seznam oznak značilk in pripadajočih dejavnikov SEO se nahaja v dodatku A.

Dobljene vrednosti p so večinoma visoke. Če postavimo mejo α pri 0.1, nam ostanejo le značilke: A6, A10, A15, A16, A17, A19, A21, A23 in A26. Za gradnjo dobrega klasifikatorja pa je to premalo.

značilka	ReliefF	p
A1	-0.0470	0.6810
A2	-0.0650	0.8240
A3	-0.0307	0.8650
A4	-0.0530	0.7460
A5	-0.0215	0.7110
A6	0.0063	**0.0870**
A7	0.0170	0.3400
A8	-0.0030	0.4570
A9	-0.0260	0.6100
A10	0.0770	**0.0940**
A11	0.0460	0.2540
A12	-0.0680	0.8220
A13	-0.0660	0.8080
A14	-0.0530	0.8210
A15	0.0890	**0.0600**
A16	0.1530	**0.0110**
A17	0.1130	**0.0580**
A18	0.0230	0.1350
A19	0.1370	**0.0650**
A20	0.0820	0.1210
A21	0.0780	**0.0230**
A22	0.0254	0.1180
A23	0.1325	**0.0100**
A24	-0.0195	0.5050
A25	-0.0267	0.6170
A26	0.0638	**0.0060**
A27	-0.0282	0.7620

Tabela 4.5: Ocene značilk

Poglavje 5

Zaključek

Z diplomskim delom smo želeli preveriti, ali lahko s pomočjo metod strojnega učenja zgradimo tak model, ki bi dovolj dobro napovedoval uvrstitve strani na seznamu rezultatov iskalnika. Nalogo smo poenostavili in preverili, kako dober bi bil klasifikator, ki bi ločeval med tem, ali se stran uvrsti med prvih 10 rezultatov iskanja (tj. na prvo stran iskalnika). Najboljše rezultate je dosegla metoda z naključnimi gozdovi. Klasifikacijska točnost je bila 0.8343, AUC pa je znašal 0.6543. Ker so take vrednosti nezadostne za kvaliteten klasifikator, smo poskusili še z uporabo klasifikatorja SVM^{rank}, ki zna razvrščati strani po vrstnem redu. Tudi ta klasifikator ni presegel točnosti naključnih gozdov.

Morebitno izboljšanje rezultatov smo videli v izločitvi nekoristnih značilk. Zato smo jih s pomočjo permutacijskega testa ocenili, vendar smo ugotovili, da le 9 značilk izhaja iz nenaključnih podatkov, kar pa je premalo za izgradnjo dobrega klasifikatorja.

Izhodiščne testne podatke bi lahko izboljšali tako, da bi bolje izbrali ključne besede. Mogoče bi se bolje obnesle zelo specifične ključne besede ali pa bi izbrali ključne besede, sestavljene iz dveh ali več besed. Testne podatke bi lahko razširili z večjim naborom ključnih besed in tako povečali število učnih primerov.

Iz dobljenih rezultatov lahko sklepamo, da uporabljene značilke niso dovolj dobre za izgradnjo kvalitetnega učnega modela. Lahko bi poskušali najti še druge dejavnike in jih izmeriti ter vključiti v raziskavo, vendar je ta naloga zahtevna. Za iskalnik Google ne obstaja uradna specifikacija dejavnikov, ki jih iskalnik upošteva pri razvrščanju rezultatov. Najdemo pa lahko številne

neuradne vire (spletne dnevnike in članke), kjer so navedeni tudi kakšni drugi dejavniki, za katere avtorji smatrajo, da bi lahko bili pomembni pri razvrščanju rezultatov. Take dejavnike bi bilo tudi zanimivo vključiti v raziskavo, vendar je med njimi veliko takih, ki jih ne moremo izmeriti s podatki, ki so nam dostopni (npr. redno osveževanje vsebine strani, starost povezav, frekvenca obiskov strani, čas zadrževanja uporabnika na strani itd.).

Iz množice značilk, ki so merljive in ki smo jih vključili v to raziskavo pa nismo dobili dobrih rezultatov. Iz tega lahko zaključimo, da iskalnik Google uporablja še druge, kompleksnejše dejavnike za določanje vrstnega reda rezultatov iskanja.

Literatura

[1] C. Sherman, *Google Power: Unleash the Full Potential of Google*, Emeryville: McGraw-Hill/Osborne, 2005.

[2] E. Enge et al, *The Art of SEO: Mastering Search Engine Optimization*, Sebastopol: O'Reilly Media, 2009.

[3] I. Kononenko, M. Kukar, *Machine Learning and Data Mining: Introduction to Principles and Algorithms*, Chichester: Horwood Publishing Limited, 2007.

[4] J. Grappone, G. Couzin, *Search Engine Optimization: An Hour a Day*, Indianapolis: Wiley Publishing, 2006.

[5] R. Rolih, *Trženje s pomočjo spletnih iskalnikov: kako so spletni iskalniki spremenili nakupno vedenje in kako lahko podjetja to izkoristijo pri trženju*, Ljubljana: GV Založba, 2007.

[6] (2009) J. West, *Google Best Practices Guide*. Dostopno na: http://www.googlebestpractices.com/google-best-practices.html

[7] (2010) Search Engine Market Share. Dostopno na: http://marketshare.hitslink.com/search-engine-market-share.aspx?qprid=4

[8] (2009) Search Engine Ranking Factors. Dostopno na: http://www.seomoz.org/article/search-ranking-factors

[9] Damping factor in Google page ranking. Dostopno na: http://www.personal.psu.edu/users/j/x/jxz203/lin/Lin_pub/2006_ASMBI_1.pdf

[10] Learning to rank. Dostopno na: http://en.wikipedia.org/wiki/Learning_to_rank

[11] Limit Distribution for the Weighted Rank Correlation Coefficient Dostopno na:
http://www.ine.pt/revstat/pdf/rs060301.pdf

[12] Request Google's Page-rank Programmatically. Dostopno na:
http://www.codeproject.com/KB/aspnet/Google_Pagerank.aspx

[13] SGMLReader. Dostopno na:
http://developer.mindtouch.com/SgmlReader

[14] SVM^{rank}: Support Vector Machine for Ranking. Dostopno na:
http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[15] Tab-delimited and similar formats. Dostopno na:
http://orange.biolab.si/doc/reference/tabdelimited.htm

[16] The Anatomy of a Large-Scale Hypertextual Web Search Engine. Dostopno na:
http://infolab.stanford.edu/ backrub/google.html

Dodatek A

Seznam značilk

značilka	dejavnik
A1	ključna beseda v naslovu strani
A2	ključna beseda na začetku naslova strani
A3	število vseh besed v naslovu strani
A4	ključna beseda v meta opisu strani
A5	število besed v meta opisu strani
A6	število ključnih besed v vsebini strani
A7	ključna beseda v znački H1
A8	ključna beseda na začetku značke H1
A9	ključna beseda v ostalih značkah H
A10	ključna beseda v atributu alt značke <IMG>
A11	ključna beseda v imenu slike
A12	ključna beseda v značkah <B>in <STRONG>
A13	ključna beseda v značkah <I>in <EM>
A14	ključna beseda v značkah <LI>
A15	ključna beseda v besedilu notranje povezave
A16	ključna beseda v besedilu izhodne povezave
A17	ključna beseda v imenu domene
A18	ključna beseda v imenu poddomene
A19	ključna beseda v nazivu strani v URL
A20	ključna beseda v nazivu mape v URL
A21	uporaba notranjih okvirjev
A22	število napak po standardih W3C
A23	število pokvarjenih povezav
A24	starost domene
A25	namenski naslov IP
A26	ključna beseda v zunanji povezavi
A27	PageRank