UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Jurinčič

PRIDOBIVANJE KONTEKSTA Z UPORABO SPLETNIH BRSKALNIKOV

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU RAČUNALNIŠTVA IN INFORMATIKE, SMER INFORMATIKA

Mentor: izr. prof. dr. Marko Bajec

Ljubljana, 2011

Rezultati diplomskega dela so intelektualna lastnina Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje Fakultete za računalništvo in informatiko ter mentorja.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01771/2011
Datum: 01.09.2011

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

Kandidat: **MARKO JURINČIČ**

Naslov: **PRIDOBIVANJE KONTEKSTA Z UPORABO SPLETNIH BRSKALNIKOV**

ACQUISITION OF CONTEXT WITH THE USE OF WEB BROWSERS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Današnje inteligentne aplikacije morajo za svoje delovanje beležiti in uporabljati kontekst uporabnika. Definirajte kontekst in preglejte obstoječe metode za njegovo beleženje. Izberite si najbolj informativen pristop, zasnujte in izdelajte orodje, ki bo znalo pametno zbirati ustrezne podatke. Orodje testiranje na manjšem naboru uporabnikov in prikažite rezultate. Rezultati morajo omogočati nadaljnje analize.

Mentor:

prof. dr. Marko Bajec

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Marko Jurinčič,

z vpisno številko 63060098,

sem avtor diplomskega dela z naslovom:

PRIDOBIVANJE KONTEKSTA Z UPORABO SPLETNIH BRSKALNIKOV

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom izr. prof. dr. Marka Bajca
- 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 8.12.2011 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se izr. prof. dr. Marku Bajcu za mentorstvo pri izdelavi diplomske naloge. Zahvaljujem se tudi univ. dipl. ing Slavku Žitniku za napotke, prijaznost, razpoložljivost ter za pomoč, ki mi jo je nudil v času pisanja diplomske naloge. Posebna zahvala gre tudi mojim staršem in vsem ostalim, ki so mi v času študija in pisanja te naloge stali ob strani in me podpirali.

Kazalo

Povzetek **1**

Abstract **2**

1 Uvod **3**

2 Kontekst **5**
2.1 Zaznavanje konteksta 8
2.2 Kontekstno zavedajoče se računalništvo 8
2.3 Uporabniški profil 10

3 Analiza obstoječih primerov **11**
3.1 Firefox Piggy Bank 11
3.2 The Lemur Project 13
3.3 SemWeb 15
3.4 Semantični spletni brskalnik 18
3.5 Beleženje premikov miške pri opravljanju preproste naloge . . . 19
3.6 Pregled izbranih rešitev 20

4 Opis problema **22**

5 Izbira spletnega brskalnika **23**
5.1 Google Chrome 27
5.1.1 Opis izdelave vtičnika 28
5.2 Mozilla Firefox 32
5.2.1 Datoteka install.rdf 34
5.2.2 Razširitev brskalnika z XUL 34
5.2.3 Registracija paketa 35
5.2.4 Ostali razlogi za odločitev 35

6 Zasnova in izdelava lastne rešitve **38**
6.1 Programski jezik . . . 39
6.2 Podatkovna baza . . . 40
6.3 Izbrani parametri . . . 41
6.3.1 Tabela “users” . . . 41
6.3.2 Tabela “tab_obj” . . . 41
6.3.3 Tabela “gotopage” . . . 43
6.3.4 Tabela “sizes” . . . 44
6.3.5 Tabela “coordinates” . . . 45
6.3.6 Tabela “advertisements” . . . 45
6.3.7 Tabela “elementclicks” . . . 46
6.3.8 Tabela “inputboxes” . . . 47
6.3.9 Tabela “shortcuts” . . . 47
6.3.10 Tabela “previouseurl” . . . 48
6.3.11 Tabela “time” . . . 48
6.3.12 Tabela “scrolling” . . . 49
6.3.13 Tabela “links” . . . 49
6.3.14 Tabela “vocabulary” . . . 50
6.3.15 Tabela “keywords” . . . 51

7 Rezultati **53**
7.1 Analiza . . . 53
7.1.1 Primerjava uporabnikov . . . 64

8 Zaključek in nadaljnje delo **66**

A Namestitev in osnovna uporaba vtičnika **68**

Seznam slik **70**
Literatura **72**

Seznam uporabljenih kratic in simbolov

SQL - Structured Query Language

URL - Uniform Resource Locator or Universal Resource Locator

RDF - Resource Description Framework

WWW - The World Wide Web

URI - Uniform Resource Identifier

XML - Extensible Markup Language

HTML - HyperText Markup Language

XMLS - Extensible Markup Language Schemas

CSS - Cascading Style Sheets

JSON - JavaScript Object Notation

API - Application Programming Interface

PHP - PHP Hypertext Preprocessor

W3C - World Wide Web Consortium

XPCOM - Cross Platform Component Object Model

XUL - XML User Interface Language

DOM - Document Object Model

HTTP - Hypertext Transfer Protocol

DTD - Document Type Definition

API - Application Programming Interface

Povzetek

Cilj diplomske naloge je izboljšanje pridobivanja konteksta o uporabnikih. Delo se prične s pregledom izbrane literature in primerov o pridobivanju konteksta v računalništvu. Skozi povezovanje teorije in prakse iz različnih disciplin postavi teoretično in tehnično osnovo za izbiro ustreznega načina pridobivanja konteksta. Rezultati pregleda so pokazali, da se splača uporabiti spletne brskalnike in kot prikaz tega je razvit vtičnik. Pri iskanju ustreznega spletnega brskalnika sem v izbiro vključil peterico brskalnikov z največjim tržnim deležem. Poraba sistemskih virov in podpora za razvoj vtičnikov sta izbiro omejila na Mozillo Firefox in Google Chrome. Nadaljnji pregled prednosti in slabosti je izločil še Mozillo Firefox, predvsem zaradi potrebe po ponovnem zagonu brskalnika pri testiranju vtičnika. Vtičnik je sestavljen iz različnih funkcij, s katerimi se beleži podatke o uporabnikih in spletnih straneh, ki jih obiskujejo. Pri izbiri podatkov sem upošteval dela in raziskave drugih avtorjev, ter prednosti, ki bi jih posamezni podatki in njihove povezave lahko prinesle pri uporabi pridobljenega konteksta. Vtičnik in shranjevanje podatkov je realizirano na način, ki omogoča dodajanje novih podatkov za beleženje. Delovanje vtičnika sem testiral en teden s šestimi uporabniki. Rezultati so pokazali nekatere zanimive vzorce kot je način poudarjanja vsebin, čas na posamezni spletni strani, obiski določenih strani vsak dan ob istem času, itd. V iskanju učinkovitejše rešitve sem tudi prišel do ugotovitve, da mora biti zajemanje in shranjevanje podatkov za uporabnika nevidno in da mora biti njegova interakcija z vtičnikom najmanj možna.

Ključne besede:

spletni brskalnik, kontekst, beleženje konteksta, shranjevanje konteksta, Google chrome, uporabniška spletna izkušnja, vzorec, razvoj vtičnika

Abstract

The purpose of the diploma thesis is to improve the retrieval of users context. The work begins with a review of selected literature and examples of context retrieval. Theory and practice from various disciplines were included in order to construct a theoretical and a technical base to choose the suitable way for context retrieval. The review showed that web browsers are the best choice and to demonstrate that a plug-in was developed. In the process of choosing the adequate web browser I have included five web browsers with the biggest market share. The system resource usage and the support for plug-in development have narrowed the decision to Mozilla Firefox and Google Chrome. The further comparison of advantages and weaknesses have eliminated Mozilla Firefox, mainly because of the need to restart the browser when testing the plug-in. The plug-in consists of different functions with which the users web browsing data is recorded. In the process of defining the functions I have taken into account the work and researches of other authors as well as the benefits that the data and the connection between different data types can bring to the usage of the retrieved context. The plug-in and the data storage is developed in a way, that allows adding new types of data to be recorded. The plug-in has been tested for one week with six users. The results have shown some interesting patterns like the way to point out some content, the time spent on a single web page, daily visits of certain pages at the same time period, and so on. In the search of a more effective solution I have come to the conclusion that the data retrieval and storage has to be invisible to the user and the interaction between the two has to be as minimal as possible.

Key words:

plug-in, web browser, context, context detection, context storage, Google chrome, users web experience, pattern, plug-in development

Poglavje 1

Uvod

Svetovni splet je vir informacij z virtualno neskončnim potencialom. Potrebujemo le internetno povezavo in lahko vsi dostopamo do dokumentov in multimedijskih virov, ki se nanašajo na vse predstavljive tematike. Uspešnost svetovnega spleta je v veliki meri posledica njegove decentralizirane oblike: spletne strani gostujejo na številnih računalnikih, kjer vsak dokument lahko kaže na druge dokumente, bodisi na istem ali različnih računalnikih. Potencial spleta pa je kljub vsemu relativno neizkoriščen, saj podatki še vedno obtičijo na različnih spletnih straneh in je težavno za računalnike te informacije smiselno procesirati. Danes svetovni splet povezuje več kot 15 milijard strani. Pridobitev pomembnih informacij na spletu je zelo vroče raziskovalno področje [52]. Današnji spletni iskalniki so narejeni tako, da služijo potrebam vseh uporabnikov, neodvisno od posebnih potreb posameznikov. Vsakič, ko uporabnik išče določeno tematiko, se sreča s stotinami povezav, ki govorijo o izbranem področju. Pridobivanje uporabnih informacij je tako prepuščeno uporabniku. Potrebno je prilagoditi spletna iskanja z vključitvijo uporabnikovih interesov. Drži pa dejstvo, da imeniki (kot je Yahoo![1]) in spletni iskalniki (Google[2], Alta Vista[3]) sestavijo nekakšen uporabniški profil in tako ponujajo pomoč pri iskanju vsebin, vendar so še daleč od popolnosti. Za večino uporabnikov je iskanje prave informacije primerljivo z iskanjem igle v seniku [52]. To pomeni, da bi morali spletni brskalniki doseči stopnjo, v kateri bi lahko "razumeli"uporabnike. To lastnost imenujemo dojemanje (angl. "perception") vendar je to samo ena stran kovanca. Po drugi strani pa morajo biti spletni brskalniki (in računalniški sistemi na splošno) sposobni izkoristiti to razume-

[1] `http://www.yahoo.com`
[2] `http://www.google.com`
[3] `http://www.altavista.com`

vanje s tem, da prilagajajo njihovo obnašanje (npr. da se odzovejo in prikažejo podobne vsebine glede na podan kontekst). To imenujemo prilagajanje (angl. "adaptivity") [1]. Najprej je potrebno izbrati informacije, ki bodo omogočile brskalnikom, da preidejo na višjo stopnjo. Tu se ne moremo omejiti samo na vsebino spletne strani, ampak tudi na navade uporabnika, npr. premiki miške, kliki, čas na spletni strani, itd. in seveda te podatke ustrezno zabeležiti. Najhitrejši način za razvoj takega orodja je uporaba funkcionalnosti, ki so že vgrajene v brskalnike in možnost njihove razširljivosti z uporabo vtičnikov.

Da bi se lahko ustrezno spoprijeli s problemom je potrebno najprej definirati o čem sploh govorimo, ko pravimo, da je nekaj kontekst. Predstavil bom nekaj definicij slednjega in se skozi njihovo razlago poskusil čim bolj približati njegovemu pomenu na področju, ki ga ta naloga pokriva. V to se bom poglobil v drugem poglavju, v katerem bom poleg konteksta razložil tudi druge pojme, kot so zaznavanje konteksta, kontekstno zavedajoče se računalništvo (angl. "context-aware computing") in uporabniški profil. V tretjem poglavju bom pregledal obstoječe primere in v četrtem na kratko opisal, kako sem se lotil izdelave. V petem sledi primerjava in izbira spletnega brskalnika ter v šestem ogrodje in opis podatkov, ki jih bo vtičnik beležil. Nato preostane še analiza rezultatov, zaključne besede in nadaljnje delo.

Poglavje 2

Kontekst

Beseda kontekst izvira iz študije človeških besedil in spoznanja, da kontekst spremeni interpretacijo teksta. Vsekakor pa se moramo pri iskanju izvora te ideje vrniti za nekaj tisočletij nazaj. Eden izmed mnogih primerov zabeležene analize in interpretacije konteksta zajema dela filozofske šole Legalistov [65], ki so bili vplivni v letih 500-60 pred Kristusovim rojstvom. V Zahodnem svetu filozofije nekdo z lahkoto identificira zamisel o zaznavanju konteksta v Grški epistemologiji[1] Dolga zgodovina konteksta prinese s seboj tudi številne definicije. Da bi lažje razumeli bistvo, si nekatere izmed njih poglejmo v nadaljevanju.

Kontekst je definiran kot tisti del pogovora, ki obdaja besedo ali odlomek in lahko poudari njen pomen (angl. "the parts of a discourse that surround a word or passage and can throw light on its meaning") [30]. Definicija pravi, da kontekst obdaja in celo poudarja določene vsebine. Torej kontekst niso samo besede, ampak je tudi okolje, ki obdaja to vsebino. Brez ustrezne umestitve besede, povedi, odlomki, itd. izgubijo želeni pomen. Določene vsebine tudi poudarja, kar pomeni, da tudi, če je celotno besedilo, del konteksta, še ne pomeni, da so vse vsebine enako pomembne. Tu se je potrebno osredotočiti na tiste, ki so pomembnejše in izpustiti ostale, saj prevelika količina informacij lahko hitro postane moteča.

Obstaja pa še veliko drugih definicij konteksta. V delu, ki kot prvo uvede pojem "kontekstno zavedajoči se" Schilit in Theimer [7] govorita o kontekstu kot o lokaciji, identiteti bližnjih ljudi in objektov ter spremembi teh objektov. V podobni definiciji Brown in ostali [29] definirajo kontekst kot lokacijo,

[1] Pomeni znanje, znanost, spoznanje. Je veja filozofije, ki se ukvarja z naravo in obsegom (omejitvami) znanosti. Naslavlja naslednja vprašanja: Kaj je znanost? Kako pridobimo znanost? Kako vemo to, kar vemo? [61]

identitete ljudi okoli uporabnika, dnevni čas, sezono, temperaturo, itd. Dey [2] predstavi kontekst kot čustveno stanje uporabnika, središče pozornosti, lokacijo in orientacijo, datum in čas, objekte in ljudi v uporabnikovem okolju. Te definicije, ki definirajo "kontekst kot primer" se uporabljajo s težavo. Ko poskušamo določiti, ali je tip informacije (ki je ni v definiciji konteksta), ni jasno ali lahko uporabimo definicijo za reševanje problema ali ne.

Druge definicije preprosto predstavljajo sinonime za kontekst; npr. definicija, ki opredeljuje kontekst kot okolje ali situacijo. Nekateri si predstavljajo kontekst kot uporabniško okolje, drugi pa kot okolje aplikacije. Brown [44] pravi, da kontekst predstavlja elemente uporabniškega okolja o katerih se zaveda uporabnikov računalnik. Franklin in Flaschbart [11] ga vidita kot situacijo uporabnika. Ward in ostali [6] vidijo kontekst kot stanje okolja, ki obdaja aplikacijo, Rodden in ostali [57] ga definirajo kot nastavitve aplikacije. Hull in ostali [47] so k vsebini dodali celotno okolje z definiranjem konteksta kot odraza trenutne situacije. Kot pri definicijah kot primer, je tudi te primere definicij zelo težko uporabiti v praksi.

Definicije Schilita in ostalih [8], Deya in ostalih [4] ter Pascoeta [25] se še najbolj približajo definiciji, ki bi jo sam rad dosegel. Schilit in ostali pravijo, da so pomembni vidiki konteksta: kdo si, s kom si in katere vire imaš na razpolago. Kontekst definirajo kot stalno spreminjajoče se izvršitveno okolje. Vključujejo naslednje dele okolja:

- **računalniško okolje** - kateri procesorji so na voljo, naprave, ki so na voljo za uporabniške vnose in prikaze, omrežno kapaciteto, povezave in ceno procesiranja,
- **uporabniško okolje** - lokacija, kolekcija bližnjih oseb, socialna situacija,
- **fizično okolje** - stopnja svetlosti in motenj.

Dey in ostali definirajo kontekst kot uporabnikovo fizično, socialno, čustveno ali informacijsko stanje. Končno, Pascoe definira kontekst kot podmnožico fizičnih in konceptualnih stanj interesa določene entitete. Te definicije so preveč specifične. Kontekst je v bistvu celotna situacija, ki je relevantna aplikaciji in njenim uporabnikom. Ne moremo določiti kateri vidiki situacije so pomembni, saj se ti spreminjajo v odvisnosti od situacije. V nekaterih primerih je fizično okolje pomembno, v drugih pa brez pomena. Ravno zato so tudi zgornje definicije nepopolne.

Kontekst je vsaka informacija, katero se lahko uporabi za opis entitete. Entiteta je oseba, kraj ali subjekt, ki utegne biti relevantna za interakcijo med uporabnikom in aplikacijo, vključujoč tudi uporabnika in samo aplikacijo (angl.

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between the user and application, including the user and applications themselves")[3].

S to definicijo se že veliko bolj približamo želeni definiciji konteksta. Z razliko od zgornje (prvotne, najbolj splošne) definicije, tu govorimo že o opisu točno določene vsebine, ki je pomembna za uporabnika. Tu se srečamo tudi z uporabnikom in njegovo interakcijo z aplikacijo. Ta interakcija nastane zaradi želje uporabnika po določenih vsebinah, katere kot že povedano, obdaja kontekst. Ni pa vse v aplikaciji povezano s tem kontekstom, zato se je potrebno vprašati, kaj je. Uporabniki se med seboj razlikujejo in kar se lahko izkaže kot kontekst za enega je lahko za drugega nepomembno. Zato je za vsakega posebej potrebno pridobiti kontekst, ki mu najbolj ustreza. Ne gre pa samo za vsebine, ki jih najdemo na spletni strani, ampak za veliko več. Kontekst je tudi samo obnašanje uporabnika; kako premika miško, koliko časa preživi na spletni strani, itd. O vsem tem lahko govorimo kot o interakciji.

Bolj podrobno definicijo bi težko našli, saj je pojem preveč splošen. Določeno znanje šteje kot kontekstna informacija na enem področju, na drugem pa je popolnoma zgrešeno [50]. Izraz ima dolgo zgodovino v različnih vejah računalniške znanosti, in sicer v umetni inteligenci, pridobivanju informacij, analizi slik in video posnetkov, kontekstno-občutljivi pomoči (angl. "context-sensitive help"), večopravilnemu spreminjanju konteksta (angl. "multitasking-context changing"), psihološkem zaznavanju konteksta (angl. "psychological contextual perception") itd. [13].

Efektivna uporaba kontekstnih informacij v računalniških aplikacijah je še vedno velik izziv. Različni raziskovalci so poskusili kategorizirati kontekstno zavedajoče se aplikacije in funkcije, kar vključuje zaznavanje konteksta, prilagajanje konteksta, kontekstno raziskovanje virov (angl. "contextual resource discovery") in bogatenje konteksta[2] (angl. "contextual augmentation"). Te ideje se lahko povežejo in uporabijo za prikaz informacij in servisov uporabniku, avtomatskemu izvrševanju servisov ali pa generiranju zaznamkov nad kontekstom za kasnejše beleženje [3].

Potrebno se je zavedati, da kljub definiranju konteksta še nismo rešili glavnega problema s katerim se srečamo v tej nalogi. Tu se srečamo z uporabniki svetovnega spleta in je potrebno za vsakega poiskati kontekst, torej vsebine, ki so zanj relevantne. Dejstvo je, da so človeške preference kompleksne, številne, heterogene, se spreminjajo, v nekaterih primerih si celo nasprotujejo in jih je

[2]Možnost povezovanja digitalnih vsebin z uporabniškim kontekstom.

potrebno razumeti v povezavi z uporabniškimi željami in tekočimi nalogami. Niso vse uporabniške želje relevantne v vseh situacijah.

Če želimo poiskati kontekst v dani situaciji se moramo vprašati kdo? (identiteta), kje? (lokacija), kdaj? (čas) in kaj? (kaj uporabnik dela - vrsta aktivnosti) in z uporabo teh informacij lahko določimo tudi zakaj se je za to odločil. Aplikacija sama ne more razumeti, zakaj se določena situacija zgodi, vendar z uporabo preteklih informacij o uporabniku, bi ji lahko to "povedali" in bi se prilagodila.

Informacije je potrebno na kakovosten način tudi pridobiti oziroma zaznati.

2.1 Zaznavanje konteksta

Beleženje/zaznavanje konteksta je že dalj časa ključnega pomena na različnih področjih, če naštejemo le nekatere so to: mobilne naprave, računalniški jezik (angl. "computional linguistic"), avtomatska analiza slik ali pridobivanje informacij. Preden aplikacija zazna oziroma zabeleži določene vsebine, se mora najprej zavedati, da so slednje relevantne. Na Wikipediji[3] [60] je zavedanje o kontekstu v računalniški znanosti opredeljeno kot ideja, da lahko računalniki zaznajo in posledično tudi reagirajo v odvisnosti od okolja. Definicija pravi, da bi morali računalnike pripeljati do točke, ko se bodo lahko samostojno prilagodili, če bo situacija to zahtevala. To zaznavanje mora biti narejeno samostojno, brez interakcije uporabnika, ki bi brskal po spletu brez da bi se zavedal, kaj se v notranjosti aplikacije dogaja. Aplikacija se bo tako prilagajala v skladu z uporabnikovimi akcijami. Brez ustreznega konteksta bi bilo tako prilagajanje nemogoče. Namen kontekstno zavedajoče se računalništva je torej izboljšanje uporabniške izkušnje skozi neprestano prilagajanje uporabe, dostopa, strukture in obnašanja informacij, servisov in fizičnih virov z upoštevanjem razpoložljivih informacij [27].

2.2 Kontekstno zavedajoče se računalništvo

Najprej si bomo pogledali splošne primere uporabe konteksta. Hull in ostali [47] ter Pascoe in ostali [25], [26], [37] definirajo kontekstno zavedajoče se računalništvo kot možnost, da se računalniške naprave zavedajo in zaznavajo, razlagajo in se odzovejo različnim vidikom uporabnikovega okolja in same aplikacije. Dey [2] omejuje kontekstno zavedanje na uporabniški vmesnik, kateri

[3] `http://www.wikipedia.org`

stoji nad aplikacijo. Dey in ostali [4] so začeli uvajati pojem prilagajanja s tem, da so definirali kontekstno zavedanje kot delovanje, ki vodi do avtomatizacije računalniškega sistema na osnovi znanja iz uporabniškega konteksta. Salber in ostali [12] definirajo kontekstno zavedanje kot zmožnost zagotovitve maksimalne fleksibilnosti računalniškega servisa osnovanega na realno-časnem zaznavanju konteksta.

Sledeča definicija prilagajanja konteksta velja za bolj specifično. [8] definira kontekstno zavedajoče se aplikacije kot aplikacije, ki se dinamično spreminjajo ali prilagajajo njihovo delovanje v odvisnosti od konteksta aplikacije in uporabnika. Bolj podrobno, Ryan [38] definira kontekstno zavedajoče se aplikacije kot aplikacije, ki nadzorujejo vnose iz okolja in omogočajo uporabnikom, da izberejo iz palete fizičnega in logičnega konteksta glede na njihove trenutne interese in aktivnosti. Ta definicija je omejujoča, kot prejšnja, s tem, da identificira metodo, v kateri aplikacije delujejo na podlagi konteksta. Brown [45] definira kontekstno zavedajoče se aplikacije kot aplikacije, ki avtomatsko zagotavljajo informacije in/ali se odzovejo glede na prisotni kontekst uporabnika, ki ga sistem zabeleži. Poda tudi ozek pogled na kontekstno zavedajoče se računalništvo z navedbo, da te akcije lahko zavzamejo obliko predstavitve informacij uporabnikom, izvajanje programa v odvisnosti od konteksta, ali konfiguriranje grafične postavitve v odvisnosti od konteksta. Fickas in ostali [51] definirajo okoljsko-usmerjene aplikacije, kot aplikacije, ki so vnaprej definirane oziroma uporabniško definirane smernice.

Dobro definicijo za kontekstno zavedajoče se računalništvo so podali pri [3], kjer pravijo da je sistem kontekstno zavedajoč se, če uporablja kontekst za zagotavljanje relevantnih informacij in/ali servisov za uporabnike, kjer je relevantnost odvisna od uporabnikove naloge (angl. “A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task”).

Gre za bolj splošno definicijo. Če poskusimo uporabiti zgornje definicije za vzpostavitev kontekstno-zavedajoče se aplikacije, vidimo, da ne služijo ravno svojemu namenu. Aplikacija, ki prikazuje kontekst uporabnikovega okolja uporabniku ne spreminja obnašanja, ampak je kontekstno zavedajoča se. Če uporabimo manj generalne definicije, te aplikacije ne bi bile klasificirane kot kontekstno zavedajoče se.

Zgornjo definicijo bom primerjal s prejšnjimi definicijami. Zgornja definicija je veliko bolj splošna kot tista, ki so jo predstavili Hull in ostali [47] ter Pascoe in ostali [25], [26], [37]. Avtorji od svojih kontekstno zavedajočih se sistemov zahtevajo, da zaznajo, razložijo in se odzovejo na kontekst. Zgornja definicija zahteva odgovor na kontekst, ki omogoča izvršitev detekcije in inter-

pretacije s strani drugih računalniških entitet. Od ostalih definicij se razlikuje tudi po tem, da se osredotoča na uporabnika, ne pa da se njeno zavedanje omeji le na uporabniški vmesnik [2]. Ne zahteva, da aplikacije opravljajo servise samostojno [4] in ne zahteva realno-časnega zajemanja podatkov [12].

Samo zavedanje pa je še vedno neuporabno, če se ne loči potrebe različnih uporabnikov med seboj. V delu [40] so se osredotočili ravno na vlogo integracije in personalizacije pridobljenih informacij pri razvoju uporabniških profilov. Uporabniški profil lahko razumemo tudi kot računalniško predstavitev modela uporabnika.

2.3 Uporabniški profil

Zelo pomembno vlogo v personaliziranju ima tudi generiranje uporabniških profilov, ki so nekakšen povzetek vsega, kar določen uporabnik počne na spletu. Med različnimi definicijami [39], [56], [68] se mi zdi najbolj primerna definicija Wikipedije, ki pravi, da je uporabniški profil zbirka osebnih podatkov, ki so povezani z določenim uporabnikom. Profil se tako navezuje na eksplicitni digitalni prikaz uporabnikove identitete. Vse tri definicije govorijo o uporabniškemu profilu, kot o zbirki podatkov, vendar se v drugem delu zgornje zelo dobro povzame bistvo, da ni to nič drugega, kot eksplicitni digitalni prikaz uporabnikove identitete. [56] in [39] pravita, da uporabnikov profil ne vsebuje samo osebnih podatkov, vendar tudi različne nastavitve (zaslona, aplikacij, omrežne nastavitve, itd.). Vse kar uporabnik vidi na zaslonu npr. v Windows-ih in ni vsebovano v privzetih nastavitvah se pri ponovnem zagonu pojavi, ker je shranjeno v uporabnikovem profilu. V bistvu se uporabniški profil zahteva ob vsaki prijavi (v operacijski sistem ali določeno aplikacijo) in se tako s pomočjo vsebovanih informacij, ki se pridobijo z zaznavanjem konteksta, aplikacija ustrezno prilagodi. V procesu pridobivanja informacij ima močan vpliv negotovost, saj omejeno število vnesenih nizov v spletni iskalnik ne more popolnoma opisati želje uporabnika. Uporabniški profili služijo zmanjšanju te negotovosti z uporabo večje količine informacij o uporabnikovih željah in ne samo splošne profile, ki bi jih aplikacija utegnila povezati z določeno osebo glede na pridobljene podatke iz zgodovine spletnega brskalnika. Vloga uporabniških profilov je razumevanje efekta dedovanja pri pridobivanju informacij, ko dva različna uporabnika predstavita enako zahtevo in dobita različne rezultate. Profil uporabnika se generira skozi konstantno spremljanje njegovih akcij, ki vsebujejo manjšo stopnjo negotovosti zaradi narave njegovih odločitev.

Poglavje 3

Analiza obstoječih primerov

Kontekstno zavedajoče se računalništvo je bilo prvič predstavljeno s strani Schilita in Theimerja [49] leta 1994 kot programska oprema, ki se prilagodi glede na lokacijo uporabe, zbirko bližnjih oseb in objektov ter sprememb teh objektov skozi čas. Prva raziskava na tem področju je bila izvedena v delu [48] leta 1992. Od takrat naprej je bilo narejenih veliko poizkusov in v tem poglavju si bomo pogledali nekaj obstoječih rešitev. Osredotočili se bomo na tiste, ki so jih uporabljali v spletnih brskalnikih:

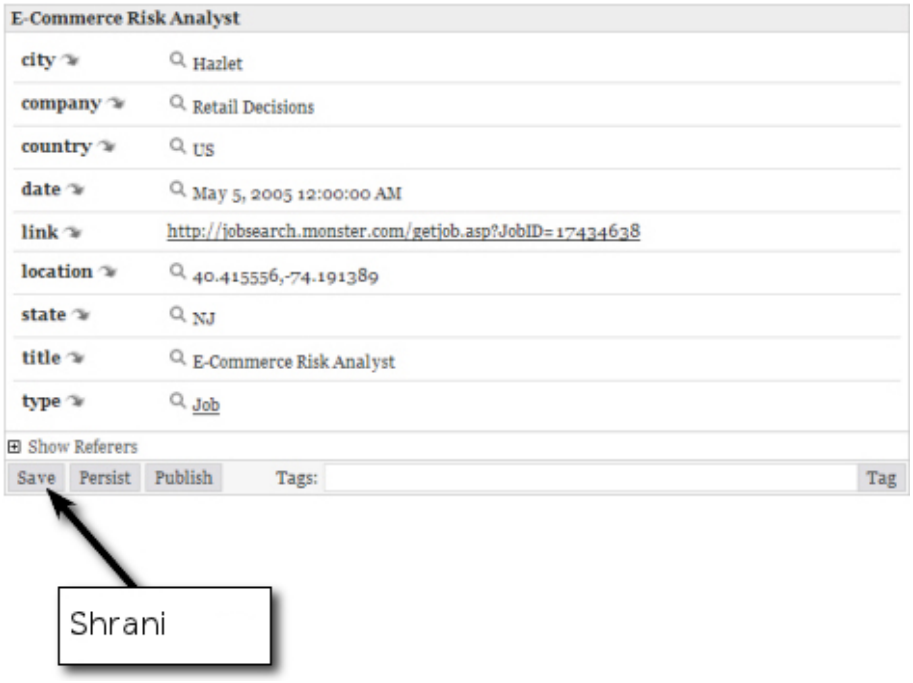
- Firefox Piggy Bank
- The Lemur Project
- SemWeb
- Semantični spletni brskalnik
- Beleženje premikov miške pri opravljanju preproste naloge

3.1 Firefox Piggy Bank

Piggy Bank je vtičnik za Mozilla Firefox[1], ki spremeni spletni strežnik v premešano[2] (angl. "mash-up") ogrodje, s tem da dovoli črpanje (angl. "extract") in kombinacijo informacij iz različnih strani.

[1] `http://www.mozilla.org/en-US/firefox/new/`

[2] Pri razvoju spletnih aplikacij je "mash-up" spletna stran ali aplikacija tista, ki uporablja kombinirane podatke, predstavitev ali funkcionalnosti iz dveh ali več virov za generiranje novih servisov. Beseda lahko pomeni tudi, hitro integracijo, pogosto z uporabo odprtokodnih

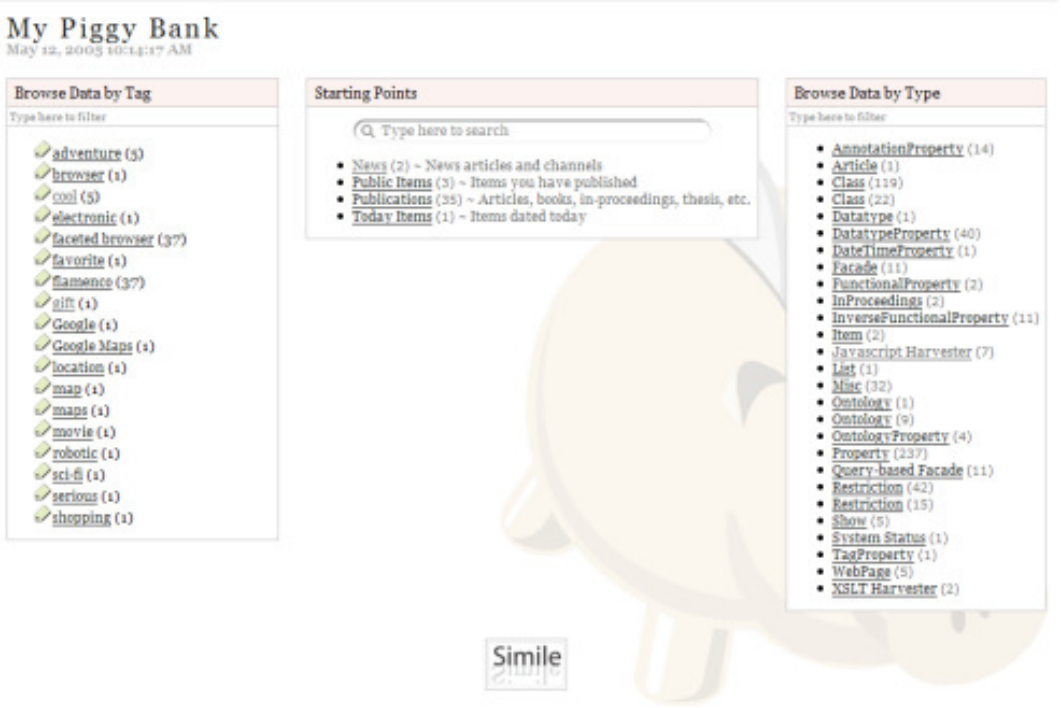


Slika 3.1: Shranjevanje podatkov v Piggy Bank-u (vir: [42]).

Omogoča tudi lokalno shranjevanje informacij za kasnejše brskanje in izmenjavo z ostalimi uporabniki vtičnika (slika 3.1). Podatke shranjuje v RDF[3] obliki. Če je spletna stran že povezana z RDF informacijami, črpanje enostavno pomeni pridobiti te informacije. V nasprotnem primeru, Piggy Bank uporablja po meri izdelano programsko kodo, katera razvozla "čiste" informacije iz formata spletne strani. Po prenosu informacij na računalnik, Piggy Bank lahko uporabi lastni uporabniški vmesnik, da omogoči brskanje skozi te informacije neodvisno od originalne spletne strani. Piggy Bank lahko pokliče Google Maps[4], da prikaže določeno lokacijo kljub temu, da osnovna spletna stran ne ponuja možnosti geografskega prikaza podatkov. Z informacijami iz različnih spletnih strani v istem podatkovnem modelu, Piggy Bank ponuja globalen pregled (slika 3.2) na vse podatke ne glede na njihov izvor [41].

aplikacij in podatkovnih virov za generacijo obogatenih rezultatov, ki niso obvezno osnovni razlog za generiranje surovih podatkov [67].

[3] "Resouce Description Framework" - RDF je družina W3C specifikacij, ki so bile v osnovi zasnovane kot metapodatkovni podatkovni model. Uporabljen mora biti kot generalna metoda za konceptualni opis ali modeliranje informacij, ki so implementirane v spletne vire, z



Slika 3.2: Pregled po kategorijah razdeljenih podatkov Piggy Banka (vir: [43]).

3.2 The Lemur Project

Lemur Project[5] je nastal kot plod sodelovanja med Centrom za Inteligentno Pridobivanje Informacij na Univerzi Massachusetts Amherst[6] in Inštitutom za Jezikovne Tehnologije na Univerzi Carnegie Mellon[7]. Razvili so Lemur Toolkit[8], ki je odprtokodno okolje za razvoj programske opreme za jezikovno modeliranje in pridobivanje informacij, ter INDRI iskalnik[9]. Zbirko orodij uporabljajo tudi pri razvoju drugih produktov, kot so: iskalniki, orodja za analizo teksta, orodne vrstice v brskalnikih in podatkovne vire na področju pridobivanja informacij. Eno izmed orodij, ki se uporablja za pridobivanje podatkov je Query Log Toolbar. Gre za vtičnik, ki teče na strani odjemalca

uporabo različnih oblik sintakse [69].

[4] `http://maps.google.com/`

[5] `http://lemurproject.org/index.php`

[6] `http://www.umass.edu/`

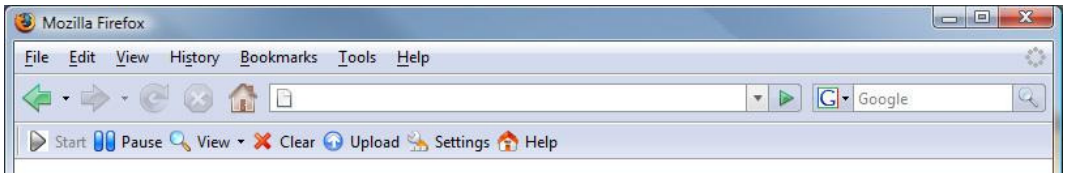
[7] `http://www.cmu.edu/index.shtml`

[8] `http://lemurproject.org/lemur/`

[9] `http://lemurproject.org/indri.php`

(angl. "client-side") in se ga lahko uporablja v brskalniku Mozilla Firefox ali Internet Explorer[10]. Shranjuje obiskane URL naslove, čase ogleda strani, kopiranje teksta, dogodke miške in rezultate iskanja spletnih iskalnikov. Ponuja tudi možnost, da se nekatere podatke prekrije (slika 3.4) (doda se jih na črno listo) in se jih lokalno zamenja z generičnim tekstom "##–##". Prikriti podatki bi lahko bili npr. telefonske številke, gesla, itd. Pri prenosu podatkov na strežnik je možno prikriti tudi spletne naslove, da ni možno vedeti kaj je uporabnik obiskal. Vtičnik je dostopen na [54].

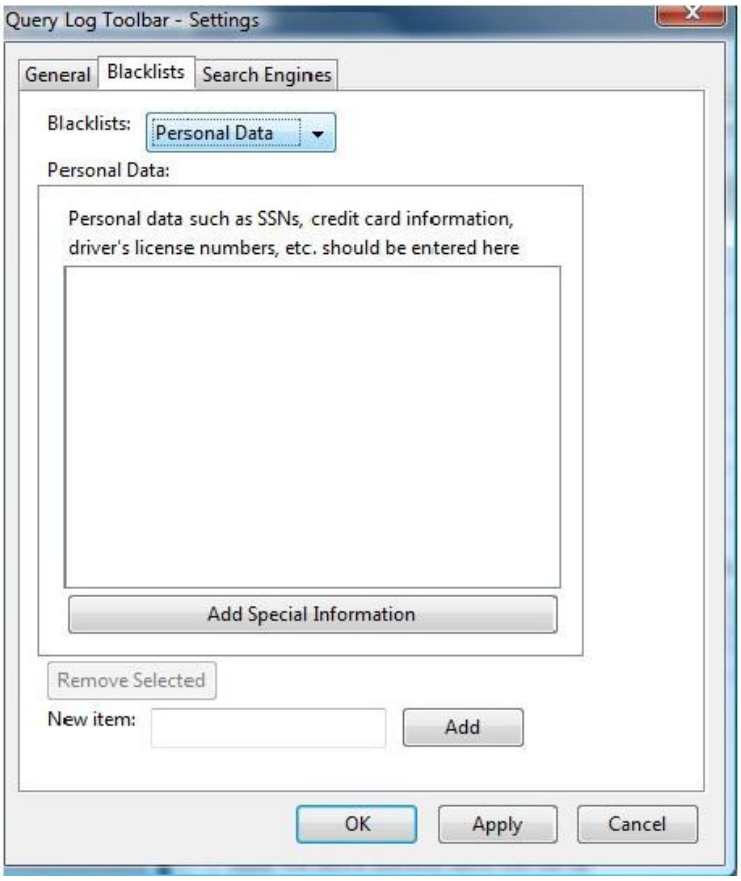
Po namestitvi se pojavi nova orodna vrstica (angl. "toolbar") (slika 3.3) kjer lahko upravljamo z vtičnikom (start, premor, pogled rezultatov, itd.). Podatke se pošlje strežniku tako, da se izbere možnost naloži (angl. "upload"). Med samim brskanjem se podatki shranjujejo na lokalni disk v log datoteke.



Slika 3.3: Orodna vrstica Query Log Toolbara (vir: [28]).

Projekt je v osnovi zelo podoben temi mojega dela, vendar se razlikuje v nekaj ključnih elementih. Če je v zgornjem primeru potrebno pritisniti tipko za prenos podatkov, se v mojem primeru pošiljajo avtomatsko, ko se okno (zavihek) zapre. Druga razlika je v tem, da se lahko izbira katere podatke se shranjuje in pošilja na strežnik. V primeru Lemur Projecta selahko uredi zapise, vendar to zahteva več časa. Zelo zanimiva je možnost prikrivanja določenih podatkov. To je tudi ena izmed potencialno najbolj problematičnih zadev v vtičniku. Ker se zaznavajo vse akcije tipkovnice, se lahko zabeležijo tudi gesla, kar bi lahko pomenilo možnost zlorabe, če uporabniki niso pozorni [53].

[10] `http://windows.microsoft.com/en-US/internet-explorer/products/ie/home`



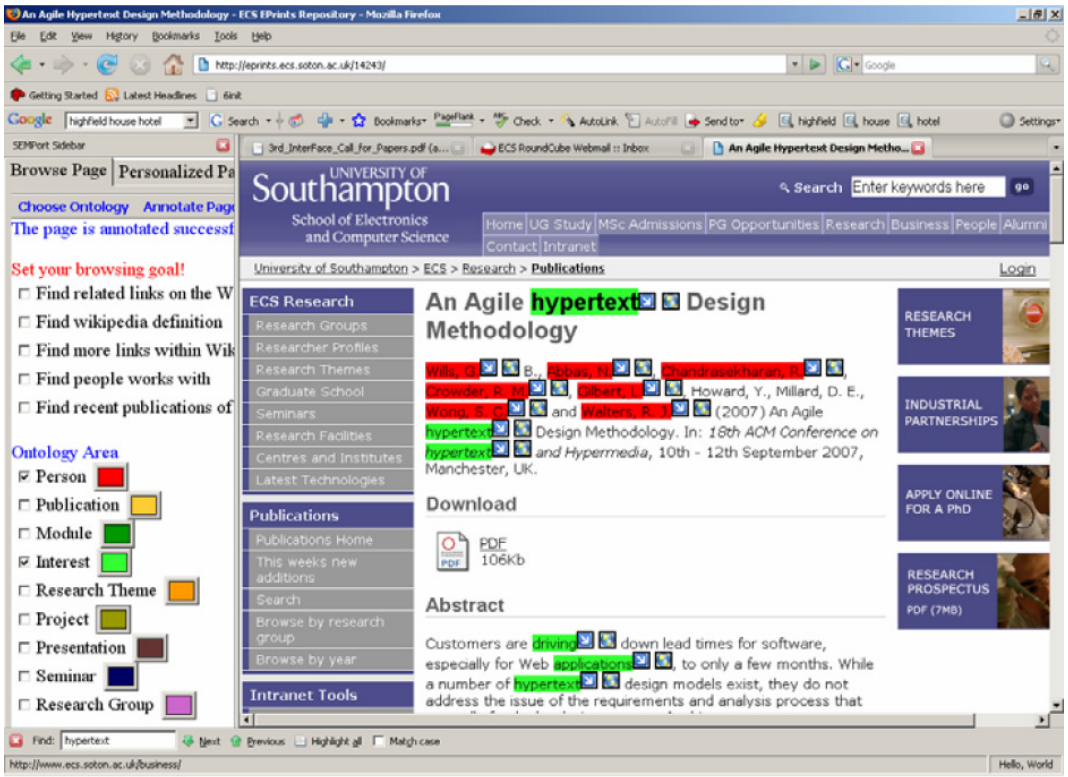
Slika 3.4: Uporabniški vmesnik v katerega se vnese podatke, ki jih uporabnik želi prikriti (vir: [28]).

3.3 SemWeb

Gre za vtičnik, ki je narejen za Mozillo Firefox. Sistem je sestavljen iz naslednjih delov:

Stranski meni za prikaz semantičnih in personaliziranih informacij

Spletni brskalnik so razširili s stranskim menijem (slika 3.5), ki zagotavlja kontekstno osnovane semantične povezave in personalizacijo za katerokoli spletno domeno. Uporabnik lahko uporablja meni za izbiro povezane podatkovne domene, notacijo spletne strani in dodajanje/poudarjanje semantičnih povezav na spletni strani. Z uporabo vgrajenih povezav uporabnik lahko zahteva semantične povezave.



Slika 3.5: Stranski meni in označene povezave na spletni strani (vir: [32]).

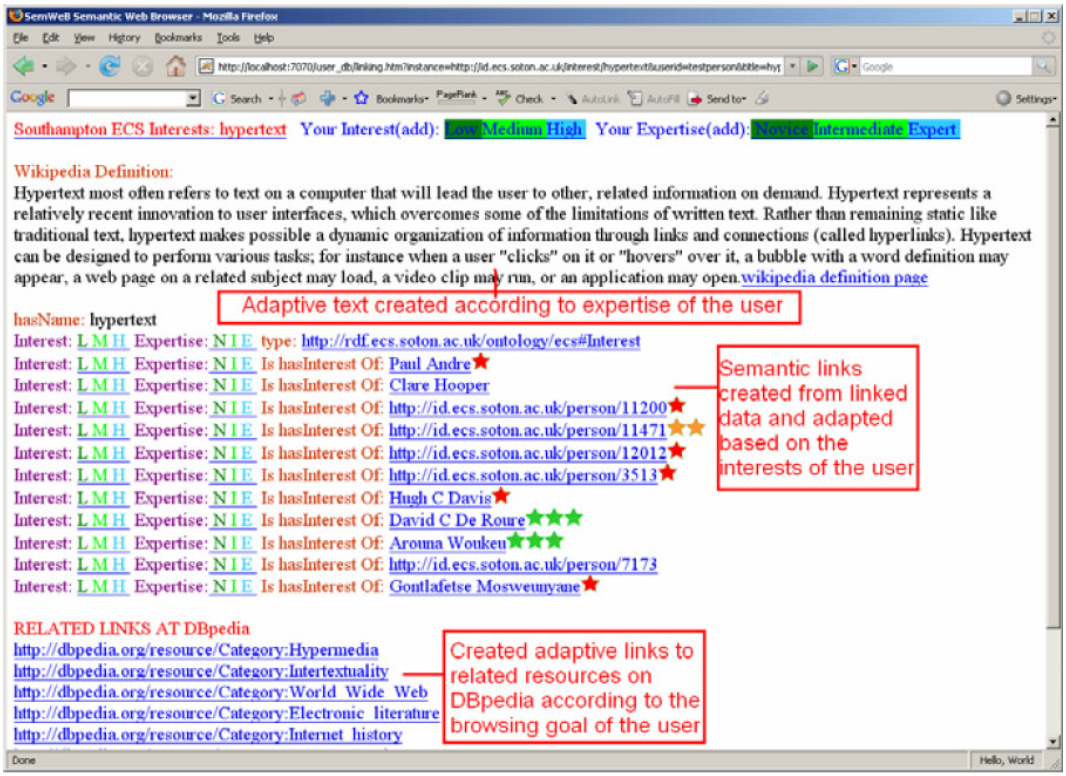
Procesiranje naravnega jezika in beleženje semantike

Za ponudbo relevantne kontekstno osnovane semantične povezave, je potrebno razumeti vsebino spletne strani. Črpanje informacij in beleženje semantike opravijo v skladu s povezano podatkovno domeno z uporabo GATE[11] okvirja (angl. "framework"). Gate okvir so razširili z iskalnim servisom za pridobivanje leksikonov, ki se ujemajo z različnimi URI-ji in generiranjem zaznamkov ter enoto za shranjevanje. Za vsako povezano podatkovno domeno, ki so jo testirali, analizirajo URI-je sorodnih podatkov in njihove leksikone za pridobivanje informacij in beleženje semantike.

[11] Gre za Javansko zbirko orodij namenjeno procesiranju nalog povezanih z naravnim jezikom [64].

Generiranje semantičnih povezav

Z uporabo zabeležene spletne strani, uporabnik lahko zahteva semantične povezave s klikom na povezave, ki so vgrajene na spletni strani. Slednje so predstavljene na novi spletni strani (slika 3.6). V primeru, da je uporabnik prijavljen, so zanj prilagojene.



Slika 3.6: Generirane semantične povezave in personalizirane vsebine (vir: [32]).

Uporabniški model za spletno-osnovano personalizacijo

V uporabniškem modelu uporabijo sedem kategorij: identifikacijo, preference, varnost, cilje brskanja, interese, strokovna znanja in izkušnje ter vedenje brskalnika. Model povezuje uporabniške profile z nesledljivimi (angl. “dereferenceable”) povezanimi podatkovnimi URI-ji, kot rezultat je uporabniški profil usmerjen proti porazdeljenim virom na spletu in se ga razširi z vsako semantično instanco.

Profiliranje uporabnikov

Uporabniki se registrirajo in prijavijo v SemWeb preko vtičnika v spletnem brskalniku. Lahko dodajajo interese do semantičnih instanc, ki jih prepoznajo na spletnih straneh in hkrati zagotavljajo znanja in izkušnje za uporabnika med brskanjem spleta. Za vsako povezano podatkovno domeno, so v stranskem meniju zagotovljeni vnaprej definirani cilji.

Prilagajanje

Obstajajo različne vrste prilagajanja, ki se razlikujejo glede na cilj brskanja, informacije iz povezanih podatkov, itd.

Primer prikazuje in deluje že na naslednji stopnji. To je obdelava, prikaz in dejanska uporaba pridobljenih podatkov. Predstavlja eno od možnosti uporabe pridobljenih podatkov. Dejansko bi se lahko s temi podatki doseglo boljše uporabniške profile, ki bi poleg informacij, na spletni strani vsebovali tudi prvine navad uporabnika. V zgoraj opisanem primeru je potrebna interakcija uporabnika, ki si mora sam ustvarjati zaznamke, kategorije, itd. [32]. Sam bi rad dosegel čim večjo avtomatizacijo in tako relativno malo dejavnikov, ki bi uporabnika utegnilo motiti pri brskanju.

3.4 Semantični spletni brskalnik

Pri tem projektu so se avtorji lotili razvoja spletnega brskalnika, s katerim bi lahko brskanje po spletu prilagajali uporabnikom.

Ko uporabnik pošlje iskani niz, bo slednji preusmerjen na strežnik, ki zajame vsebino iskanega niza z uporabo matrike izrazov (angl. “term matrix”). Efektivnost mehanizma pridobivanja je izboljšana v tolikšni meri, da strežnik vrne najboljše 3 kategorije, ki so povezane z iskanim nizom. Uporabnik lahko izbere kontekst brskanja glede na pridobljene kategorije. Izbira se znova pošlje na strežnik, kateri sedaj preusmeri poizvedbo na uporabniku najljubši spletni iskalnik. Tako se bo zmanjšalo število pridobljenih rezultatov iskalnika. Efektivnost tega bo manjša, če se bo uporabnikova pričakovana kategorija nahajala na dnu seznama. Vendar se bo med brskanjem teh kategorij spremenil seznam in rangiranje rezultatov. Prednost delovanja semantičnega spletnega brskalnika je tako:

- semantično prepoznavanje iskanega izraza s strani spletnega brskalnika,
- personalizacija iskanja - sestavijo se uporabniški profili,

- zmanjšanje števila pridobljenih rezultatov (nerelevantnih) - pri testiranjih so dosegli 50-80 % zmanjšanje pridoboljenih rezultatov,
- pravila in tabela dejstev sta zasnovana tako, da imata pameten sistem za predpomnenje - pogoste poizvedbe in rezultati se shranijo v predpomnilnik, tako se še dodatno optimizira brskanje.

Predpomnenje pogostih poizvedb se lahko prikaže za zelo dobro funkcionalnost, saj se z naraščanjem priljubljenosti pametnih mobilnih telefonov, brskanje po spletu ponovno premika na naprave, ki niso priključne na hitre internetne povezave. S tem bi se doseglo tudi manjše količine prenosa podatkov, saj se strani nebi nalagale v celoti, ampak samo spremenjeni deli.

3.5 Beleženje premikov miške pri opravljanju preproste naloge

V tem primeru so razvili spletno stran za učenje, ki beleži premike miške študentov. Študija premike tudi analizira in poskuša pridobiti vzorce vedenja, ki bi izražali interese študentov.

Premikanje miške študentov so opazovali medtem, ko so študenti reševali različne probleme. V nalogi je profesor objavil Java programski problem. Študenti so morali potrebne metode iskati po specifikacijah Jave in jih nato uporabiti pri reševanju naloge. Njihovo obnašanje ob reševanju problema je bilo povezano s premiki miške. V analizi so si pomagali tudi s starejšimi študijami, v katerih so dokazali, da je ena izmed pomembnih lastnosti spletnih strani (ki vpliva na pozornost in premike miške) delitev spletne strani na manjše razdelke/enote. To pomeni, da vsaka spletna stran vsebuje številne majhne razdelke/enote in vsak izmed njih predstavlja osnovno komponento, s katero lahko inteligentni sistemi sklepajo interese študentov glede na njihovo ravnanje z miško.

Z zaznavanjem vedenja študentov bi lahko profesorji in sistemi za učenje predlagali specifikacije in primere za pomoč študentom. Testna množica je bila sestavljena iz 28 študentov, kateri so brskali 64 minut. Pri brskanju po Java 2 spletni strani, so študentje prikazali nekatere značilne vzorce. Po analizi posnetkov in premikov miške 28-ih študentov so zaznali 5 osnovnih vzorcev:

Pregledovanje (angl. “screening”)

Študenti so se pomikali (angl. “scrolling”) po spletni strani z uporabo srednje miškine tipke in pri tem pregledali različne spletne strani v celoti, dokler niso našli iskane informacije.

Označevanje (angl. "highlighting")

Študenti so označevali tekstovna območja z miško.

Obotavljanje (angl. "Hesitation")

Študenti so prikazali obotavljanje okoli nekaterih območij na spletni strani, kar lahko pomeni, da so tekst v teh območjih brali.

Branje s pomočjo kurzorja (angl. "Pointed reading patterns")

Študenti so večkrat premikali miško po tekstovnih območjih, kar je pokazalo, da so si verjetno s kurzorjem pomagali pri branju.

Neposredno iskanje (angl. "Direct search")

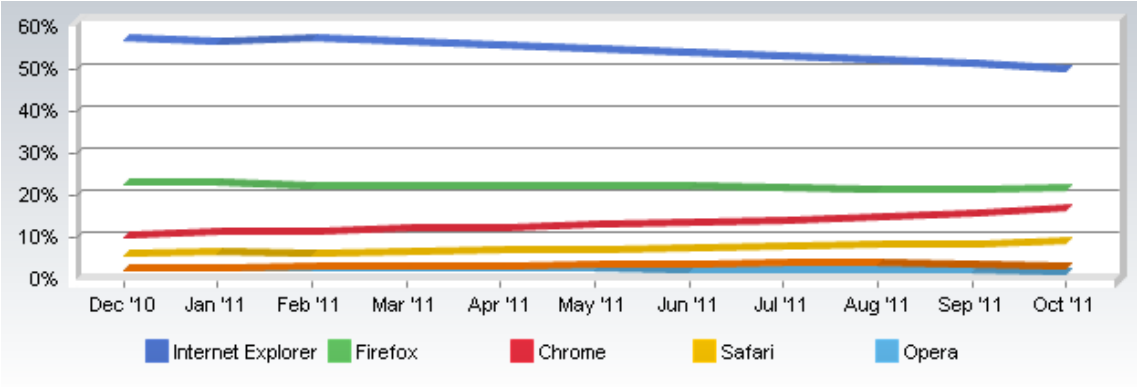
Študenti so neposredno iskali ključne besede z iskalnim menijem v brskalniku [9].

Primer se nekoliko razlikuje od ostalih, vendar lepo prikazuje, kaj vse lahko dosežemo z zaznavanjem preproste informacije kot je premikanje miške. Gre pa tudi za prikaz možnosti, ko zaznavanje podatkov ne poteka na strani odjemalca, ampak je vgrajeno kar na strežniku.

3.6 Pregled izbranih rešitev

Zgornji primeri opisujejo tri različne načine zaznavanja uporabniških interakcij s spletni brskalnikom. Gre za primere implementacije celotnega spletnega brskalnika, vtičnika in programa, ki teče na strani strežnika in deluje samo za določeno spletno stran. Razvoj novega spletnega brskalnika bi verjetno bil najslabša izbira. Med uporabniki medmrežja kroži že dovolj brskalnikov in bi se v to tekmovanje zelo težko vključili. Kljub potencialno revolucionarni zamisli, bi težko dosegli velik prehod uporabnikov. Primer tega je Internet Explorer, kateri kljub slabšim rezultatom pri primerjavah spletnih brskalnikov še vedno vodi na lestvici priljubljenosti (vendar se počasi tudi ta trend spreminja saj je novembra po dolgih letih njegov tržni delež padel pod 50 % [23]. Na to vpliva tudi rast priljubljenosti pametnih telefonov in tablic, trg katerega je Microsoft v zadnjih letih lahko samo opazoval, vendar naj bi z njihovimi novimi produkti postali močan tekmec tudi na tem področju). Bolj zanimiva je primerjava med vtičnikom in programsko kodo, ki bi beležila kontekst na določeni spletni strani. Največja prednost vtičnika je predvsem hitra in enostavna namestitev. Pri programski kodi je v večini primerov to potrebno vključiti v spletno stran. Gre za zelo preprost postopek, vendar to predstavlja

dodatno delo za lastnike spletnih strani (če sami niso razvijalci morajo poklicati razvijalca in v večini primerov tudi preplačati tako preprosto vgraditev). Težava vtičnikov je nekompatibilnost v različnih spletnih brskalnikih, kar pa ne bi veljalo za kodo, ki bi se jo vgradilo v izbrano spletno stran in bi temeljila na JavaScript programskem jeziku, katerega podpirajo vsi brskalniki.



Slika 3.7: Tržni delež spletnih brskalnikov. Internet Explorer se je po dolgih letih spustil pod 50 % (vir: [23]).

Če želimo izboljšati spletno izkušnjo za vse uporabnike se ne moremo osredotočiti le na posamezne spletne strani, vendar moramo razmišljati globalno. Glede na zgornje primere menim, da je vtičnik veliko bolj splošna izbira. Programska koda bi lahko veliko pomenila za lastnike spletne strani, s katero bi dobili podatek o potrebah in željah njihovih uporabnikov. Lastnikom spletnih strani lahko pomagamo tudi z vtičnikom. Dodana vrednost za uporabnike tu seveda ni vprašljiva, saj bi njihovo brskanje postalo učinkovitejše, ker bi jim bile ponujene samo relevantne informacije. Boljše informacije se nahajajo na spletnih straneh, katere bi tako obiskovali le uporabniki, ki bi imeli interese na spletni strani (če stran ni optimizirana za spletne iskalnike ni rečeno, da se bo prikazala v določeni poizvedbi, kljub temu, da vsebuje iskane informacije). Po izbiri načina razvoja se bom v naslednjem poglavju nekoliko bolj osredotočil na sam problem in na kratko predstavil bistvene funkcionalnosti, katere želim doseči.

Poglavje 4

Opis problema

Naloga je pridobiti kontekst in glede na primere je izdelava vtičnika[1], za enega izmed bolj popularnih spletnih brskalnikov, primerna rešitev.

Da bo vtičnik kar najbolj učinkovit, je potrebno izbrati ustrezne podatke, ki bodo ustrezno opisali spletnega uporabnika.

Pridobljeni podatki se morajo sproti shranjevati v obliki, ki je primerna za nadaljnjo obdelavo, kategoriziranje in predstavitev.

Da bi bil produkt uporaben tudi za druge brskalnike ga je potrebno razviti kar najbolj splošno (torej z minimalno uporabo funkcij in metod spletnega brskalnika). Pri uporabi mora imeti uporabnik tudi možnost izbire podatkov, katere želi zaznavati. Uporabnike je potrebno ločiti med seboj, da se lahko za vsakega ustvari uporabniški profil. Samo delovanje mora biti za uporabnika nevidno. Tudi uporaba sistemskih virova mora biti kar se da majhna. Ravno tako se uporabnika ne bo smelo obremenjevati s shranjevanjem podatkov.

[1]Vtičniki so manjši programi, kateri spremenijo ali povečajo funkcionalnosti spletnega brskalnika. Ustvari se jih s spletnimi tehnologijami kot so HMTL, JavaScript in CSS. Zanje je značilno tudi to, da imajo zelo majhne (ali pa celo nimajo) uporabniške vmesnike [18].

Poglavje 5

Izbira spletnega brskalnika

Pred začetkom implementacije je potrebno izbrati spletni brskalnik, za katerega bom izdelal osnovno različico vtičnika.

Ena izmed pomembnejših lastnosti vtičnika je neopaznost. Spletni brskalnik se zaradi tega ne sme upočasniti. To dejstvo močno vpliva na izbiro, saj je že v osnovi najboljše izbrati brskalnik z najmanjšo porabo sistemskih virov. Potrebno je razumeti, da je danes "klasičnih" deskarjev zelo malo in uporabniki na spletu gledajo filme, igrajo igrice in zaganjajo najrazličnejše aplikacije. Seveda se vtičnik zaganja tudi v teh oknih in bi ga lahko uporabnik hitro "obtožil" v primeru, da brskalnik zamrzne oziroma, da upočasni nalaganje posnetka na Youtube[1]. Za večino sodobnih brskalnikov se cena, ki jo morajo plačati pri spremljanju obogatenih vsebin, odraža v porabi spomina. Ko pride do upravljanja spomina (angl. "memory management"), razlogi za rast porabe niso preprosti. Intuitivno je, da več zavihkov kot je odprtih, večja bo poraba spomina. Vsak zavihek hrani slike, tekst in druge aktivne podatke, ki ohranjajo strani obstojne (angl. "persistent") med pregledovanjem drugih vsebin. Zraven je potrebno dodati še porabo vtičnika. Količina podatkov, ki jih vtičnik shrani ni prevelika, saj so v večini to le tekstovni podatki, vendar so uporabniki muhasti in že manjše spremembe jih lahko odvrnejo od uporabe. Seveda imata brskalnika kot sta Chrome in Firefox možnosti, kako izklopiti predpomnenje, vendar uporabnik si tega ne želi. Brez predpomnenja (angl. "cache") posnetki na YouTube se ne bi predvajali v ozadju in večina realnočasnih aplikacij ne bi delovale kot bi morale – tak način delovanja uporabljajo v Safariju[2] na mobilnih aplikacijah in je zelo moteče (še posebej v primeru, ko je internetna povezava počasna), da se pri prehodu iz enega zavihka na

[1] `www.youtube.com`

[2] `http://www.apple.com/safari/`

drugega stran ponovno nalaga. Tako delovanje je tudi razumljivo gledano iz drugega zornega kota, saj se na mobilnih telefonih vedno manj uporablja spletne brskalnike. Večina spletnih strani, ki ponuja obogatene vsebine (novice, posnetke, itd.) ima svoje aplikacije, s katerimi je omogočen hitrejši dostop in predvsem bolj prijazen pregled vsebin (spletni razvijalci ne prilagajajo strani za ekran telefona – več o temu v naslednjih poglavjih). Ko primerjamo porabo spomina na današnjih brskalnikih, moramo vzeti v zakup nekaj različnih primerov: osnovna poraba (ko brskalnik zaženemo), obremenitev (ko odpremo večje število zavihkov) in vračanje virov sistemu (ko zavihke zapiramo). Spodnje teste so naredili pri Tom's Hardware [58] in so bili objavljeni aprila letos v četrti izdaji njihovega Web Browser Grand Prix. Testi so bili opravljeni na Windows 7 operacijskem sistemu. Tu se je seveda najbolj izkazal Internet Explorer. Rezultati testa so prikazani na sliki 5.1 in so bili izvedeni na Googlovi domači strani. Explorer je zabeležil 24MB porabe, medtem ko je Chome porabil 27MB. Na drugi strani pa sta Firefox in Safari potrebovala 60 in 62 MB.

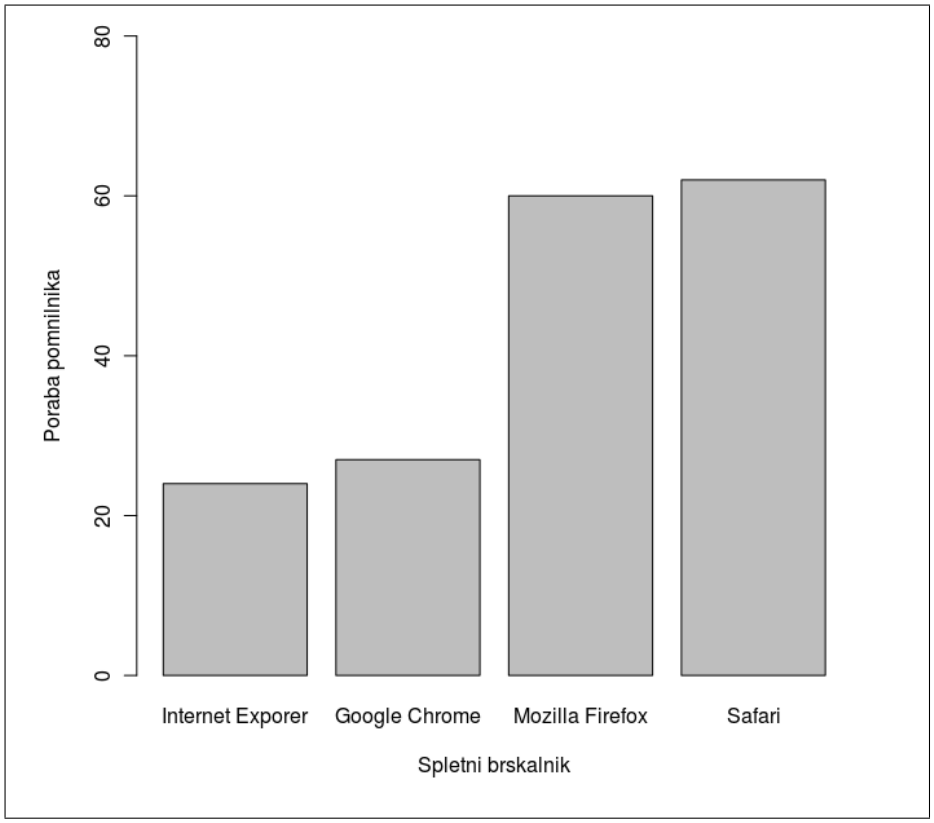
Pri [59] so izvedli podoben poizkus, vendar na operacijskem sistemu Leopard 10.6.7. Žal v primerjavi ni Internet Explorerja, vendar je vključena Opera [3]. Podatki za en zavihek so prikazani na sliki 5.2. V vodstvu je Google Chrome.

Poglejmo naprej, kako se brskalniki odzovejo pod večjo obremenitvijo. Torej če je hkrati odprtih, 40 zavihkov. Lestvica 5.2 se povsem obrne. Nova razporeditev je prikazana na sliki 5.3. Razlog za tako porabo je Chromov način odpiranja novih zavihkov v ločenih procesih. Tu pa se je potrebno vprašati, če sploh kdaj uporabljamo tako veliko število zavihkov v eni instanci brskalnika. Osebno mi postane že število med 10 in 20 skrajno nepregledno.

To pa še ni konec zgodbe. Potrebno je pogledati še, kako je s sproščanjem virov. Ta del je seveda najpomembnejši tudi za vtičnik, saj ne želimo, da bi sistem kljub zapiranju zavihkov deloval vedno počasneje – zakaj? Tu bi še enkrat rad poudaril, da splošni uporabnik spletnega brskalnika ne bo šel iskati dejanskega razloga za počasnost, ampak bo poskusil najprej odstraniti zadeve, ki so bile nazadnje nameščene. Poglejmo, odprli smo 40 zavihkov in jih zaprli 39. V tem primeru porabe pomnilnika (slika 5.4) se prepozna moč Google Chroma, saj je vračanje virov pri ostalih veliko počasnejše. Seveda se po kakšnih 5 minutah pri vseh brskalnikih skoraj vsi viri povrnejo, vendar 5 minut upočasnjenega delovanja računalnika je v današnjem času veliko preveč.

Da bi dobil podroben vpogled v porabo, sem se zadnjega testiranja lotil

[3] `http://www.opera.com/`



Slika 5.1: Primerjava porabe pomnilnika spletnih brskalnikov. Test so izvedli pri Tom's Hardware [58].

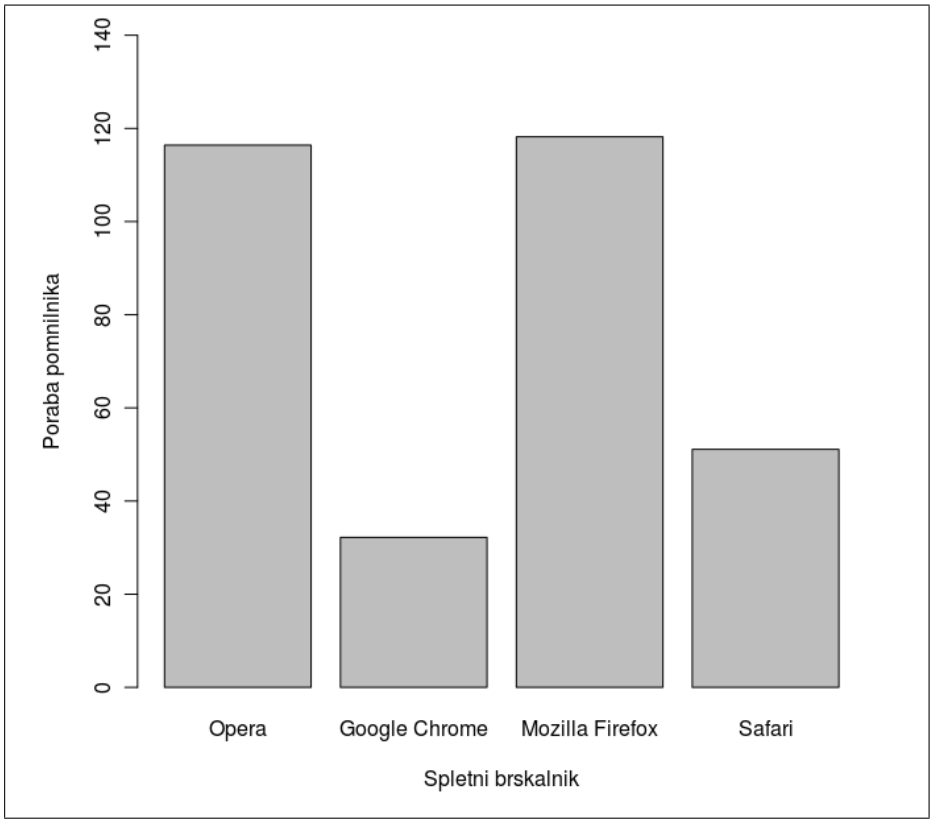
sam in ga izvedel na Ubuntu[4] (11.10 – Oneiric) operacijskem sistemu. Nekatere brskalnike je potrebno namestiti in zaganjati s pomočjo Wine[5], saj niso namenjeni Linux platformam. To sta Safari in Internet Explorer, kateri je še nekoliko bolj problematičen, saj je trenutno podprta samo verzija 7. Za merjenje porabe, sem uporabil (v Ubuntu vgrajeni) System monitor. Prvi poskus je bil izveden za merjenje porabe pri enem zavihku (slika 5.5)(`www.google.com`):

Safari in Internet Explorer že za zagon porabita preveč časa. Zato ju bom izpustil iz nadaljnjega merjenja. Na splošno se aplikacije preko Wine zaganjajo veliko časa in delujejo občutno počasneje kot na drugih operacijskih sistemih. Sledil je test s 40 zavihki (slika 5.6) – 40 krat odprta stran www.google.com:

Po hitrem zapiranju in vračanju virov sistemu pa je rezultat prikazan na

[4] `https://www.ubuntu.si/`

[5] `https://help.ubuntu.com/community/Wine`

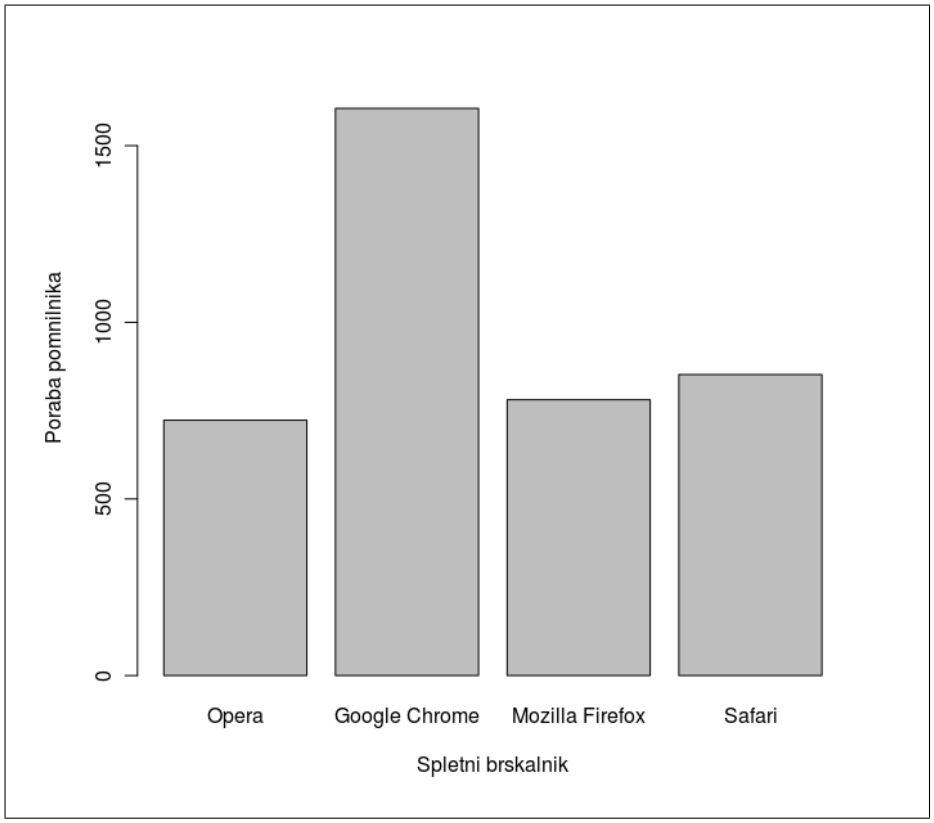


Slika 5.2: Primerjava porabe pomnilnika spletnih brskalnikov za en zavihek. Test so izvedli pri [59].

sliki 5.7:

Tu pa preseneča Google Chrome, ki pri več zavihkih ne porabi veliko pomnilnika, pa tudi vire vrne hitreje. Taka razlika je verjetno nastala zaradi novih različic spletnih brskalnikov, saj so bili zgornji testi narejeni aprila 2011, v tem času pa se je veliko spremenilo. Testi, ki sem jih sam izvedel, so bili narejeni z uporabo najnovejše opreme. Pogled na razlike, ki so nastale v tako kratkem času priča o temu, da pri Googlu veliko vlagajo v razvoj svojega brskalnika, kar je velik plus v njegov prid.

Ker bi bilo bolj podrobno primerjanje vseh brskalnikov nepotrebno, se bom osredotočil samo na dva. Ker Safari in Internet Explorer nista podprta na vseh operacijskih sistemih, ju bom izločil. Ostanejo trije – Chrome, Firefox in Opera. Tu je potreben še pogled na razširjenost spletnih brskalnikov med uporabniki, saj ne bi imelo pomena razvijati vtičnika, katerega bi uporabljalo le manjše število uporabnikov.



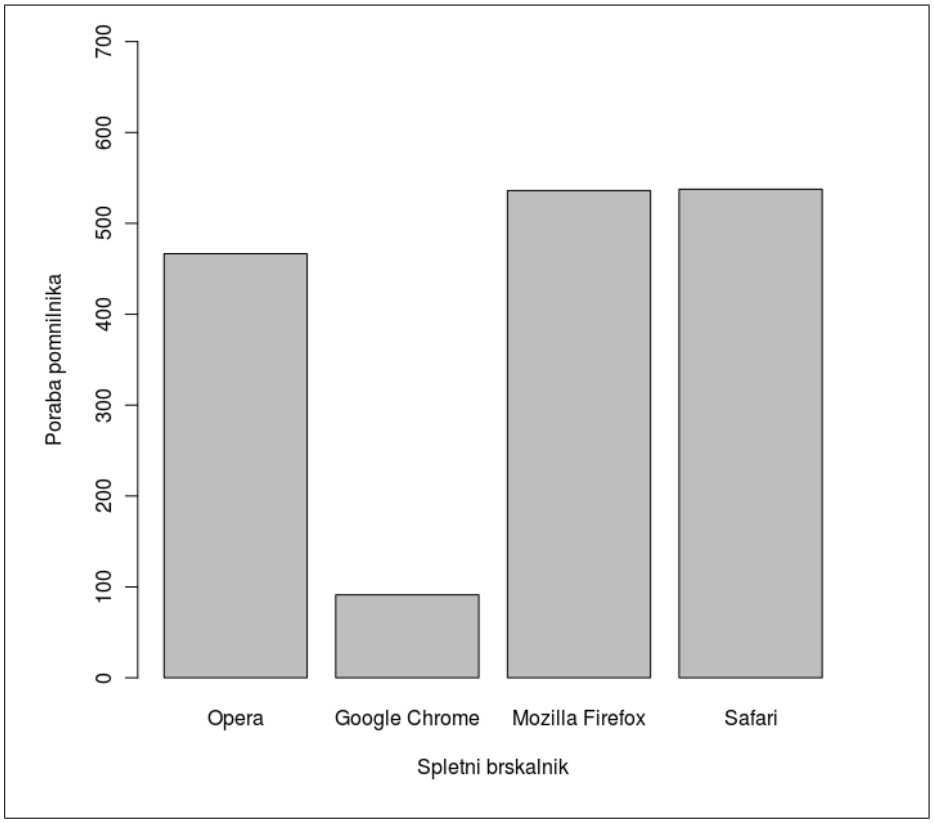
Slika 5.3: Primerjava porabe pomnilnika Spletnih brskalnikov pri 40 zavihkih. Test so izvedli pri [59].

Najbolj razširjen med spletnimi brskalniki je Internet Explorer, kateremu sledita Firefox in na tretjem mestu Chrome. Se pa trend spreminja in število uporabnikov Google Chroma neprestano raste in "jemlje" uporabnike svojima bolj uspešnima tekmecema. Po pogledu na te podatke bo iz primerjave izločena Opera.

5.1 Google Chrome

Google Chrome je spletni brskalnik, katerega razvijajo pri Google in temelji na WebKitu[6]. Prva testna različica je bila izdana 2 septembra 2008 za Windows operacijske sisteme. 11 decembra pa ji je sledila prva javna in stabilna

[6]WebKit je "layout engine" s katerim se omogoča spletnim brskalnikom, da prevajajo (angl. "render") spletne strani.



Slika 5.4: Primerjava porabe pomnilnika spletnih brskalnikov pri zapiranju zavihkov in vračanju virov. Test so izvedli pri [59].

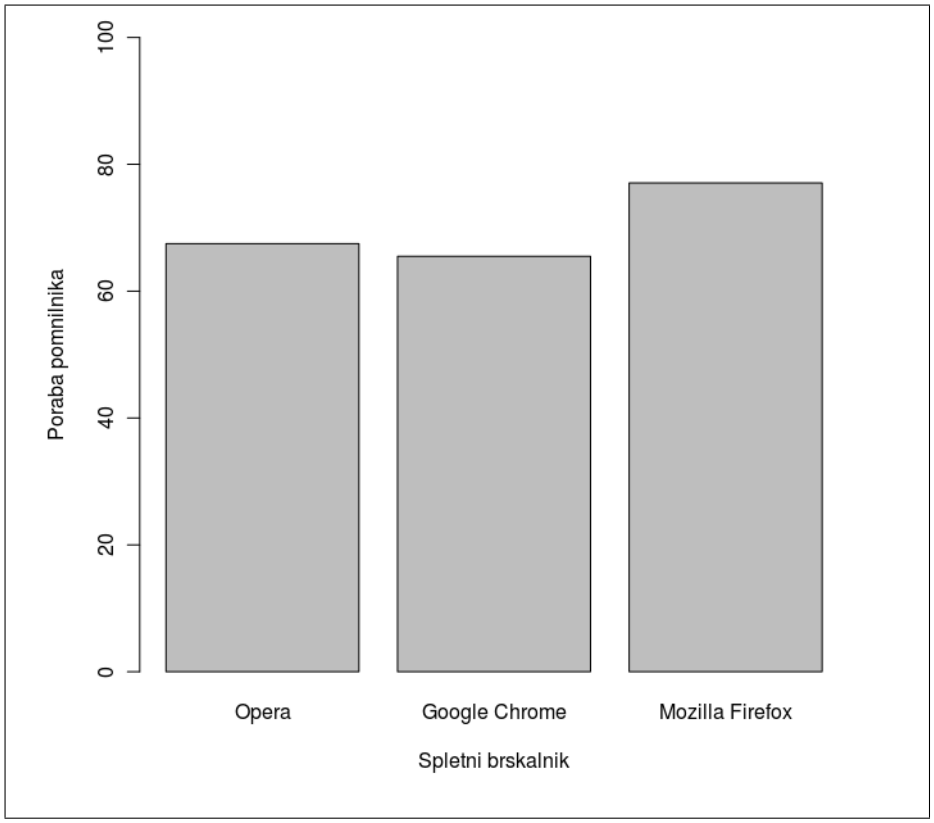
verzija. Ime izhaja iz poimenovanja okna grafičnega vmesnika spletnih brskalnikov (angl. "Chrome"). V svoji kratki zgodovini se je Chrome hitro povzpel na tretje mesto (25 % deleža - september 2011) na lestvici priljubljenosti spletnih brskalnikov [63].

V nadaljevanju sledi opis izdelave vtičnika.

5.1.1 Opis izdelave vtičnika

Najprej je potrebno povedati Chromu nekaj podatkov o samem vtičniku: ime, privzete ikone, skupaj z vsemi pravicami, ki so potrebne za dostop do funkcij brskalnika, itd. To naredimo v "manifest" datoteki, katera temelji na JSON[7] formatu. Vtičniki lahko izvajajo različne naloge, kot so prikazovanje opomni-

[7] `http://www.json.org/`

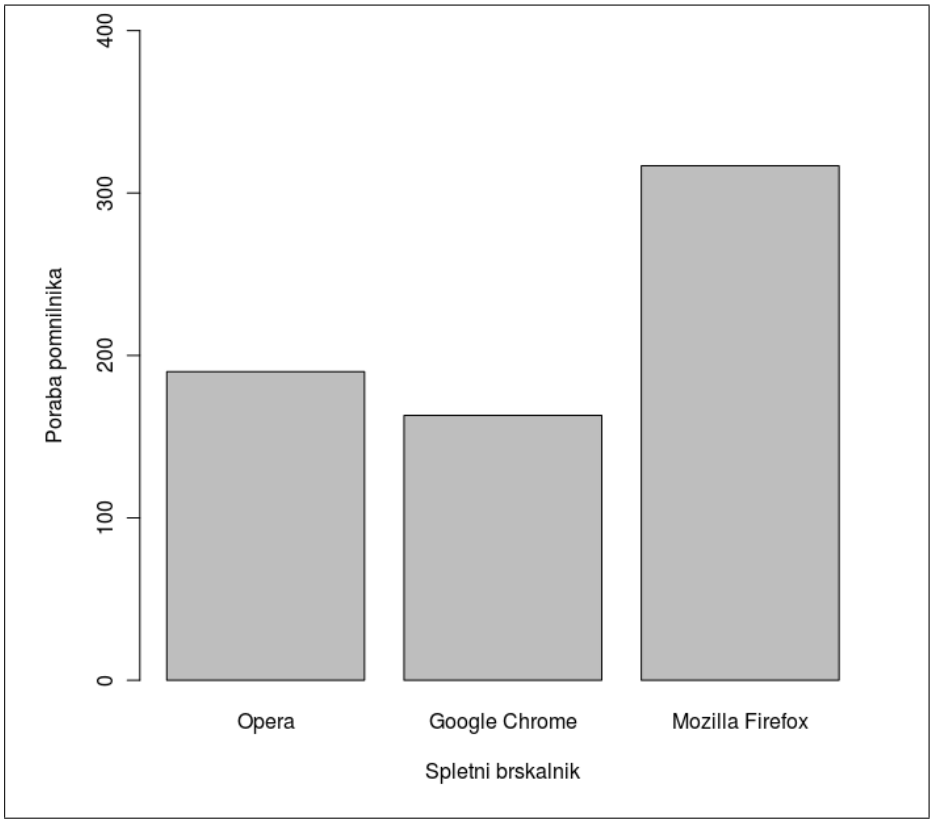


Slika 5.5: Primerjava porabe pomnilnika spletnih brskalnikov pri enem zavihku. Testirano na Ubuntu operacijskem sistemu.

kov, spreminjanje tem, ipd. (spremeni se lahko celoten izgled spletne strani). Obstajajo pa tudi akcije brskalnika, kar v bistvu pomeni npr. dodatek gumba v orodno vrstico, katerega obnašanje je neodvisno od prikazane strani (kot bi bila še ena funkcija brskalnika). Razne akcije definiramo v manifest datoteki:

```
"browser_action":{
   - "default_icon": "icon.png",
   - "default_popup": "popup.html"
   }
```

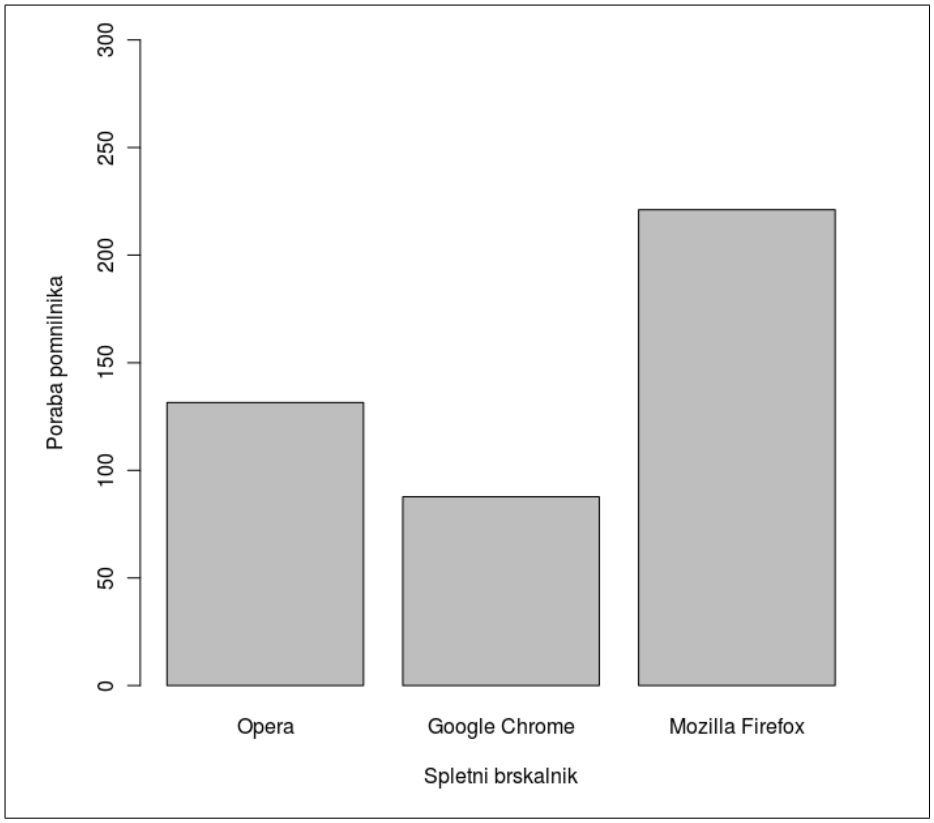
V zgornjem primeru smo definirali ikono, ki se pri oknu prikaže in stran, ki se pri tem odpre. Sedaj sledi razvoj funkcionalnosti. Implementacija pojavnega (angl. “popup”) okna je enaka kot, da bi razvijali katerokoli spletno stran. Razlika se pozna samo pri komunikaciji z ostalimi deli vtičnika. Za razvoj so pomembne tudi vsebinske skripte (angl. “content scripts”). To so



Slika 5.6: Primerjava porabe pomnilnika spletnih brskalnikov pri 40 zavihkih. Testirano na Ubuntu operacijskem sistemu.

JavaScript datoteke. Njihova koda se vgradi neposredno v spletno stran in tako lahko pridobiva oziroma spreminja informacije. Njihova uporaba se ravno tako definira v .json datoteki:

```
"content_scripts": [
  {
    "js": ["jquery-1.6.1.min.js","tabobject.js",
    "content.js","getmetadata.js"]
  }
]
```



Slika 5.7: Primerjava porabe pomnilnika spletnih brskalnikov pri vračanju virov. Testirano na Ubuntu operacijskem sistemu.

Zelo pomembno je definirati pravice, ki jih ima vtičnik. To storimo ravno tako v .json datoteki npr.

```
"permissions": ["cookies","bookmarks","tabs",
"http://*/*","https://*/*"],
```

V tem primeru je definirano, da lahko vtičnik dostopa do piškotkov, zaznamkov in dogodkov povezanih z zavihki (vse to v sklopu brskalnika) in da bo deloval na naslovih, ki se pričnejo s HTTP in HTTPS.

Poleg JSON notacije je potrebno za razvoj preproste aplikacije poznati še JavaScript in HTML.

Največja pomanjkljivost vtičnikov za Chrome je, da ne podpirajo Jave. Veliko se lahko naredi s pomočjo JavaScripta, HTML-ja in XML-ja, vendar bi bil razvoj veliko hitrejši in lažji, če bi se lahko vključilo kakšno Javansko

knjižnico. Problem je v tem, da se Jave ne more zagnati iz chrome-extension:// shem. Google tudi nima namena tega popraviti, saj meni, da bi to morali storiti na strani Jave. Možna rešitev je gostovanje Java programskega vmesnika (angl. "appleta") na strežniku. Več pomankljivosti je dosegljivih na [17].

5.2 Mozilla Firefox

Mozilla Firefox je zastonjski in odprtokodni spletni brskalnik, ki izhaja iz Mozillove zbirke aplikacij (angl. "Mozilla Application Suite"). Z njim opravlja Mozilla Corporation[8]. Trenutno zaseda drugo mesto na lestvici priljubljenosti spletnih brskalnikov z nekaj več kot 25 % deležem. Za prikaz spletnih strani uporablja Gecko[9] "layout engine".

Mozilla ponuja okolje v katerem je možno preprosto in hitro razvijati vtičnike [33], gre pa še za beta različico, zato se bom osredotočil na klasičen razvoj. Prednost okolja pa je predvsem hitro testiranje.

Firefoxovi vtičniki so zgoščene (angl. "Zip") datoteke s kratico xpi (z zanimivo izgovorjavo zippy), primer vsebine mape:

install.rdf

Splošne informacije o vtičniku.

chrome.manifest

Tu so definirani vsi elementi uporabniškega vmesnika okna aplikacije, ki so izven vsebinskega območja okna. Primeri elementov, ki so običajno del kroma (angl. "chrome") so: orodne vrstice, meniji, nalagalne vrstice (angl. "progress bars") in naslovi vrstic okna [34].

chrome/content/

Glavna datoteka za opis vsebine okna. Njen avtor je ponudnik vsebin in je lahko katerikoli tip datoteke, ki ga lahko Mozilla prevede in prikaže. Pogosto je to XUL datoteka, saj je v osnovi narejena za opis vsebine oken in dialogov. JavaScript datoteke, ki definirajo uporabniški vmesnik so tudi vsebovane v paketu vsebin, kot tudi večina XLB zavezujočih datotek.

[8] `http://www.mozilla.org/foundation/moco.html`

[9] Gecko je zastonjski in odprtokodni "layout engine", ki je zasnovan za podporo odprtim spletnim standardom in ga uporabljajo različne aplikacije za prikaz spletnih strani in v nekaterih primerih tudi same uporabniške vmesnike.

chrome/icons/default/*

Privzete ikone vtičnika.

chrome/locale/*

Sledljive aplikacije držijo vse svoje podatke pri lokalnih ponudnikih. To omogoča prevajalnikom, da vgradijo različne krom pakete za prevod aplikacije, brez da bi spreminjali ostalo izvorno kodo. Glavni obliki sledljivih datotek sta DTD datoteke in Javanske datoteke značilnosti.

Defaults/preferences/*.js

Privzete datoteke se uporabljajo za sejanje uporabniških profilov in bi se morale nahajati v defaults/ pod vtičnikovim korenom (če pogledamo hierarhijo datotek). Privzete značilnostne .js datoteke se morajo nahajati v defaults/preferences/. Če so vnesene na prava mesta, jih bo Firefox samodejno prepoznal in naložil v svoj preferenčni sistem, ko se zažene in tako omogoči uporabo priljubljenih aplikacij[10] (angl. "Preferences API"). Brez težav se lahko ustvari tudi opcijsko okno, ki temelji na XUL in omogoča uporabniku spreminjanje nastavitev. Veliko aplikacij ima dodatna orodja za super uporabnike, about:config, katera se lahko uporabi za spreminjanje nastavitev brez uporabniškega vmesnika. Primer "preference" datoteke:

```
pref("extensions.sample.username","Joe");
pref("extensions.sample.sort",2);
pref("extensions.sample.showAdvanced",true);
```

plugins

Razni dodatki za vtičnik, ki uporabljajo npr. Javo.

/components/*

Firefox podpira XPCOM komponente v vtičnikih. S tem se lahko zelo preprosto ustvari lastne komponente v JavaScript in C++ jezikih (ob uporabi Gecko SDK) Vse .js ali .dll datoteke shranimo v components/ direktorij in se nato avtomatsko registrirajo ob prvem zagonu vtičnika.

[10]Preferences API omogoča shranjevanje in branje besed, številk, logičnih spremenljivk in reference do priljubljenih datotek. Je relativno enostaven za uporabo in omogoča številne uporabniške in razvijalske funkcije. Vključuje podporo za privzete želje, uporabniško prepisovanje preko user.js in zaklepanje.

Lastne XPCOM komponente je možno zaganjati tudi s pomočjo ukazne vrstice (to tehniko se lahko uporabi za zagon vtičnika kot aplikacije).

5.2.1 Datoteka install.rdf

Za razvijalce vtičnikov je priporočljivo ustvariti nov uporabniški profil, da bi se izognili poslabšanju delovanja brskalnika in izgubo podatkov.

Najprej se zapiše v install.rdf. Primer enostavne datoteke:

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <!-- properties -->
  </Description>
</RDF>
```

Skratka, vsebuje informacije, ki identificirajo vtičnik: kdo je avtor, kje se lahko najde več informacij o njem, s katerimi verzijami aplikacij je kompatibilen, kako se ga posodobi, itd.

5.2.2 Razširitev brskalnika z XUL

Firefoxov uporabniški vmesnik je sestavljen iz XUL-ja in JavaScritpa. XUL je pravopisno XML, ki zagotavlja uporabniškemu vmesniku gradnike kot so gumbi, meniji, orodne vrstice, drevesa, itd. Same uporabniške akcije so nanje povezane z JavaScriptom. Da bi razširili brskalnik, moramo spremeniti dele uporabniškega vmesnika z dodajanjem spremenjenih gradnikov. Slednje dodamo z vnosom novih XUL DOM elementov v okno brskalnika. Spreminjamo jih z uporabo skriptnega jezika in vezanjem na dogodke. Osnovna XUL datoteka (na kateri temelji Firefox uporaniški vmesnik) se nahaja v

```
$namestitveni_direktorij_firefox-a/chrome/browser-jar.
```

Za dodajanje novih elementov je potrebno samo razširiti ta dokument (spiše se dodatne dokumente in se v njih definira, na katero mesto se npr. gumb doda).

XUL datoteke so del kromovih paketov – to je skupina komponent uporabniškega vmesnika, katere se naloži preko chrome:// URIja. Namesto, da bi se brskalnik nalagal iz diska z uporabo file:// URI (saj se lokacija Firefoxa

na računalniku lahko razlikuje od platforme do platforme), so pri Mozilli razvili rešitev za generiranje URI-jev za XUL vsebine s katerimi je seznanjena aplikacija.

5.2.3 Registracija paketa

Potrebno je registrirati "Overlay" datoteko, saj mora Firefox združiti ustvarjeno XUL datoteko s Firefoxovo osnovno. Zato je potrebno dodati naslednjo vrstico:

```
overlay chrome://browser/content/browser.xul
chrome://sample/content/sample.xul
```

Za testiranje je potrebno najprej povedati Firefoxu kje se vtičnik nahaja in nato se bo naložil vsakič, ko se brskalnik ponovno naloži.

Ko je osnova za vtičnik narejena se lahko naprej generira nova okna in pogovorne okvirje kot ločene .xul datoteke, ustvari nove funkcionalnosti z implementacijo uporabniških akcij v .js datotekah, z uporabo DOM metod za manipuliranje elementov uporabniškega vmesnika. Omogočeno je tudi dodajanje slogov v .css datotekah, dodajanje slik, spreminjanje barv, itd.

5.2.4 Ostali razlogi za odločitev

Za končno odločitev si poglejmo še nekaj primerjav med brskalnikoma. Bistvo je, da Firefox brez vtičnikov ni nič drugega kot preprost brskalnik in je velika mera njegove priljubljenosti odvisna ravno od vtičnikov. Na drugi strani pa je Google Chrome preces hitrejši, vendar je število vtičnikov zanj občutno manjše zato se veliko uporabnikov ne odloči za zamenjavo.

Enostavnost namestitve

Ena od prednosti Google Chroma je, da za namestitev vtičnikov in tem, ni potrebno ponovno zagnati brskalnika. Na drugi strani pa je to potrebno narediti pri uporabi Firefoxa. Pri razvoju je lahko to zelo moteče, še posebej, ker ob dolgem delovanju računalnika rabijo aplikacije več časa, da se zaženejo.

Performance brskalnika pri uporabi vtičnikov

Že od same predstavitve Google Chroma je bilo veliko uporabnikov skeptičnih glede počasnega delovanja brskalnika v primeru namestitve vtičnika. Pokazalo se je, da so bile bojazni neupravičene, saj Crome zaganja vse

zavihke in vtičnike v ločenih procesih in tako jim je onemogočeno, da vplivajo na celoten brskalnik. Firefox pa je znan po njegovi požrešnosti v primeru, da pride do namestitve prevelikega števila vtičnikov.

Izbira vtičnikov in funkcij

Firefox ima ogromno bazo razvijalcev in število vtičnikov je veliko večje kot pri Chromu. Dodatno k temu, se pri Firefoxu najde veliko več kompleksnih in funkcijsko bogatih vtičnikov.

Upravljanje z vtičniki

Oba brskalnika imata všečne uporabniške vmesnike za urejanje vtičnikov. Firefox pregleduje za posodobitve ob vsakem ponovnem zagonu, na drugi strani pa Chrome samodejno posodobi vtičnike. Samodejno posodabljanje je lahko moteče za nekatere uporabnike, saj nimajo vsi želje po novih verzijah programov.

Varnost

Google Chrome poganja vtičnike v ločenih procesih in okoljih (angl. "separate process in a sandbox environment"), tako da težava z enim ne bi smela vplivati na ostale. Firefox nima te funkcionalnosti, zato so morali v preteklosti nekatere vtičnike onemogočiti zaradi varnostnih težav [55].

Finančna podpora

Največji delež finančne podpore pridobiva Mozilla od Googla. Pogodbo sta podjetji že dvakrat podaljšali, vendar se je to nazadnje zgodilo leta 2008, ko je Chrome prišel na trg. Novembra 2011 se pogodba izteče in vprašanje je, ali se bosta podjetji odločili za ponovno podaljšanje. Poznavalci že dalj časa širijo govorice o vseh možnih razlogih, zakaj se Google ne bo odločil za podaljšanje pogodbe z njihovimi tekmeci. Z odstranitvijo Firefoxa iz dirke spletnih brskalnikov, bi si lahko priborili velik delež Firefoxovih uporabnikov in postali resen kandidat za odvzem vodilne vloge Internet Explorerju. Potrebno je razumeti, da bi se brez Googlovega financiranja Mozilla znašla v resnih finančnih težavah. 86 % od 104 USD milijonov vrednega poslovanja temelji ravno na tej osnovi. Brez tega privzetega denarja bi se Mozilla hitro skrčila v 15 milijonov dolarjev vredno podjetje, ki bi veliko težje tekmovalo z ostalimi spletnimi velikani. Taka poteza bi se lahko izkazala za veliko napako iz dveh razlogov. Prvi je ta, da je Firefox drugi najbolj razširjen spletni brskalnik in iz tega vidika tudi največji tekmec Internet Explorerju. Je tudi edini

brskalnik, ki je podprt na skoraj vseh operacijskih sistemih, kar pomeni, da bi bil njegov odmik zelo počasen, pa tudi vsi milijoni uporabnikov ne bi tako hitro zapustili priljubljenega brskalnika in preskočili na Chrome. Drugi razlog je ta, da je Firefox za Google tudi vir prihodka. Ko Firefox uporablja iskanje (kot privzet iskalnik ima seveda Googlovega), spletni gigant pridobi iz tega nekaj drobiža (oglaševanje). Zato so se pripravljeni odreči deležu trga, za ohranitev privilegiranega položaja njihovega iskalnika. Kljub prekinitvi pogodbe pa Mozilla verjetno ne bi potrebovala veliko časa, da bi pridobila podporo pri drugih virih, saj je veliko podjetij za katere 85 milijonov ne predstavlja prevelike vrednosti npr. Microsoft [24].

Glede na primerjavo in razlike se bom odločil za Google Chrome. Veliko mi pomeni hitrost testiranja. Ponovno zaganjanje celotnega brskalnika bi vzelo preveč časa (še posebej pri Mozilli, ki po zaprtju aplikacije potrebuje še nekaj časa da dokončno konča proces). Zaradi pomanjkanja Jave se bom verjetno moral odreči vključitvi zaznavanja posnetkov in zvoka. Obstajajo drugi način, kako to rešiti, vendar bi bil razvoj preveč dolgotrajen.

Sledi opis zasnove in uporabljene tehnologije pri razvoju.

Poglavje 6

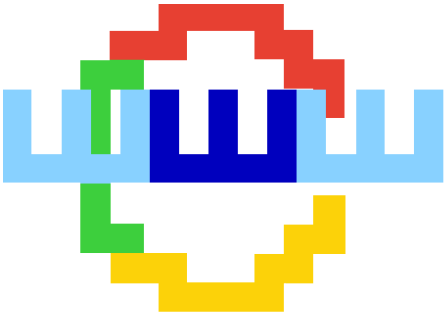
Zasnova in izdelava lastne rešitve

Vtičnik za Google Chrome je stisnjena skupina datotek - HTML, CSS, JavaScript, slik in vsega ostalega, kar se potrebuje za dodajanje funkcionalnosti brskalniku Google Chrome. V osnovi gre za spletno stran, ki lahko uporablja vse API-je, ki jih brskalnik zagotavlja spletnim stranem od XMLHttpRequestov do JSON-a do HTML5.

Vtičniki lahko komunicirajo z drugimi spletnimi stranmi z uporabo vsebinskih skript (angl. "content script") ali navzkrižnih (angl. "cross-origin") XMLHttpRequests. Sodelujejo lahko tudi programsko s funkcijami brskalnika (zavihki, zaznamki, itd.).

Veliko vtičnikov je dodanih uporabniškemu vmesniku Google Chroma z akcijami brskalnika in akcijami strani (vsak lahko ima eno ali drugo akcijo). Akcijo brskalnika je boljše izbrati, ko je vtičnik relevanten za več strani. Akcijo strani pa, ko se ikona vtičnika prikaže ali skrije v odvisnosti od strani. Seveda se lahko uporabniški vmesnik doda tudi kot vnos v meni, opcijsko stran ali z uporabo vsebinskih skript, ki spreminjajo vsebino strani. V tej nalogi sem za uporabniški vmesnik uporabi akcijo brskalnika. Ob kliku na ikono (slika 6.1) se odpre okno, v katerem se lahko izbira informacije, ki jih vtičnik nato zaznava na spletni strani [19].

Ostale funkcionalnosti so realizirane preko vsebinskih skript in "background" strani.



Slika 6.1: Ikona vtičnika. Pojavi se v zgornjem desnem kotu brskalnika in odpre okno z nastavitvami.

6.1 Programski jezik

Večji del vtičnika je realiziran kot vsebinska skripta, saj je za delovanje potrebna interakcija s spletno stranjo. Vsebinska skripta je JavaScript koda, ki se izvede v kontekstu v brskalniku naložene spletne strani. Skripto si lahko predstavljamo kot del naložene strani, ne pa kot del vtičnika iz katerega izhaja.

Za shranjevanje podatkov sem uporabil php programsko kodo v povezavi z MySql podatkovno bazo. Ob zaprtju okna se preko Ajaxa vzpostavi komunikacija s .php aplikacijo, kateri vtičnik pošlje vse pridobljene podatke. Te podatke spletna aplikacija sprejme, predela in shrani v podatkovno bazo. Ker je velikost poslanih podatkov omejena, se v primeru prekoračitve komunikacijo vzpostavi večkrat. Za komunikacijo v nasprotno smer pa se uporablja XML datoteke, katere generira strežnik in jih pošlje vtičniku, kateri iz njih prebere potrebne podatke. Za uporabniški vmesnik in "background" stran se uporabljajo HTML5 in CSS tehnologiji. Za manifest datoteko se uporablja JSON notacijo.

Komunikacija različnih delov vtičnika je realizirana preko "cross-origin" XMLHttpRequest-ov. Tu je potrebno opozoriti, da je možnost vsebinskih skript za pošiljanje zahtev do strežnikov bila dodana v verziji 13 Google Chroma. Pred tem, vsebinska skripta ni mogla direktno generirati zahtev, vendar je morala sporočilo poslati sorodnemu vtičniku, ki je nato zahtevo poslal naprej [20].

Med brskanjem po spletni strani se podatki shranjujejo v objekt imenovan tab_obj. To lahko predstavlja nevarnost, saj se podatki shranjujejo v pomnilnik in bi lahko v primeru prevelikega števila zabeleženih podatkov, prišlo do zapolnitve slednjega. Ta scenarij je zelo malo verjeten, saj so podatki izključno besede oziroma posamezni znaki in skupaj redko presežejo 1MB.

6.2 Podatkovna baza

Ob zaprtju okna ali zavihka se podatki pošljejo na strežnik, kateri jih obdela in spravi v MySql podatkovno bazo. Povezava s strežnikom je potrebna zaradi JavaScripta, saj programska koda teče na računalniku odjemalca (angl. "client") in je onemogočeno shranjevati podatke na lokalni napravi.

Google Chrome je prvi brskalnik, ki omogoča shranjevanje podatkov tudi na strani odjemalca s t.i. "sandbox-om". Gre za virtualni prostor, ki ga vtičnik zasede na računalniku uporabnika in je tako obvarovan. Do njega lahko dostopa samo vtičnik in sam ne more dostopati do drugih delov diska. Velikost prostora na disku, ki ga zasede, ni omejena in se ga definira ob deklaraciji.

Za implementacijo se nisem odločil, ker ima nekaj pomanjkljivosti. Kot že prej omenjeno lahko do njega dostopa samo vtičnik. Tudi imena se ne da izbirati in ga računalnik avtomatsko poimenuje glede na ime okna, za katerega je bil generiran (za vsako okno, ki je odprto v brskalniku se generira nov "sandbox"). Uporabilo bi se lahko za razbremenitev pomnilnika, v primeru, da bi se v vtičnik dodalo snemanje posnetkov uporabnikovega vedenja. Sestava podatkovne baze je preprosta in je prikazana v spodnjem diagramu (6.2).

V nadaljevanju sledi še opis vsebine tabel (zabeleženih podatkov).

6.3 Izbrani parametri

V prejšnjih poglavjih sem govoril o kontekstu in poskusili določiti, kaj to je in kako bi lahko s tem pripomogli k spletni izkušnji uporabnikov. Naloga vtičnika je zaznavati in beležiti ustrezen kontekst, zato se to poglavje osredotoča ravno na podatke, za katere po pregledu drugih virov in lastni presoji menim, da bi bili pomembni in uporabni za nadaljnje izračune in raziskave. Podatke bom predstavil po posameznih tabelah (kot so shranjeni v podatkovni bazi) in pri vsakem dodal, zakaj bi bil uporaben.

6.3.1 Tabela "users"

Za generiranje uporabniškega profila vtičnik zahteva prijavo. Prijava je klasična, z uporabo uporabniškega imena in gesla.

id Unikatni identifikator uporabnika.

username Uporabniško ime uporabnika.

password Geslo uporabnika.

Prijava z uporabniškim imenom in geslom je nujno potrebna, saj bi se drugače težko razlikovalo uporabnike. Uporabnike bi bilo možno razlikovati, če bi imeli na razpolago velike količine podatkov za vsakega in bi tako na podlagi statistike ustvarili profile in jih povezali s posameznimi uporabniki. V tem primeru se lahko zelo dobro približamo določitvi uporabnika, vendar še vedno obstaja možnost napake. Bolj preprosto bi bilo to realizirati z uporabo Internet Explorerja, z namestitvijo ActiveX kontrol. Z njihovo uporabo lahko preko brskalnika dostopamo do raznih sistemskih informacij (dostop do njih je v drugih primerih zaradi varnostnih razlogov onemogočen). S temi podatki bi lahko profili bolj natančno definirali in ciljali uporabnika.

6.3.2 Tabela "tab_obj"

Entitetni tip tab_obj je referenčna tabela za vse ostale. Vanjo se shranjujejo osnovni podatki o spletnih straneh.

id - vsaki spletni strani je dodeljen unikatni identifikator. Potreben je za povezavo z ostalimi tabelami.

userid - unikatni identifikator uporabnika vtičnika.

url - spletni naslov spletne strani. V primeru, da se spletni naslov strani spremeni, se jo bo v tabelo vneslo kot novo spletno stran, saj preverjanje in primerjanje za take primere ni realizirano.

visits - koliko krat je uporabnik obiskal spletno stran. S tem lahko določimo, katera spletna stran je uporabniku najbolj priljubljena in to upoštevamo pri določanju ključnih besed in rangiranju slednjih (podatki pridobljeni iz te spletne strani bi lahko imeli večjo težo kot ostali). Ob vsakem obisku se število poveča.

pagetitle - naslov spletne strani, pridobljen iz HTML značke (angl. "tag") "title".

useragent - spletni brskalnik in operacijski sistem uporabnika - ta podatek je lahko zelo uporaben v primeru določanja uporabniških identifikacijskih številk brez pomoči prijave z uporabniškim imenom in geslom (kakor je to trenutno realizirano) - več o tem v nadaljevanju. Tu sem predpostavil, da uporabniki skozi čas ne spreminjajo operacijskega sistema in verzije brskalnika. Bolj učinkovito bi bilo ustvariti novo tabelo za shranjevanje te vrednosti, v katero bi se vpisovalo v primeru, da je prišlo do spremembe. Podatek bi bil zanimiv v primeru, da uporabnik po npr. posodobitvi spletnega brskalnika začne obiskovati nove spletne strani, saj so v novi verziji uredili npr. delovanje flasha.

type - tip spletne strani - pridobljen iz metapodatkov - če avtorji spletne strani ustrezno vnašajo metapodatke (bolj kot ustrezno je pomembno, da vnesejo podatke, ki dejansko predstavljajo spletno stran) lahko hitro in učinkovito združimo spletne strani v skupine, katere povezuje podobna tematika. Tako lahko na preprost način ustvarimo kategorije in splošne uporabniške profile (Amazon[1] primer, ko napiše, da osebe, ki so prebrale to knjigo so se zanimale tudi za naslednje knjige).

time - vrednost predstavlja čas v dnevu, ko je uporabnik obiskal spletno stran. Času je v nadaljevanju namenjena lastna tabela, vendar se mi je vseeno zdelo primerno vpisati to vrednost, saj predstavlja čas, ko se je uporabnik prvič povezal na to spletno stran. Ta informacija je zanimiva, saj v povezavi z ostalimi podatki iz tabele lahko izvemo ali se je uporabnik začel zanimati za novo področje, s katerim se v preteklosti ni ukvarjal (npr. php programer je videl lestvico najbogatejših Slovencev

[1] `www.amazon.com`

in se glede na številke, ki so jih ustvarili pri Outfit 7[2] odločil, da bo začel programirati v Objective C).

6.3.3 Tabela "gotopage"

Kako je uporabnik prišel na spletno stran.

pageid - identifikacijska številka spletne strani s katero so podatki povezani.

how - podatek lahko zavzema štiri različne vrednosti in predstavlja način s katerim je uporabnik prišel na spletno stran. Vrednosti so: "favorites" (spletno stran ima med priljubljenimi v spletnem brskalniku),"search engine" (na spletno stran je prišel preko enega izmed bolj priljubljenih spletnih iskalnikov - naslov iskalnika se primerja z vsebino tabele iskalnikov),"web page" (na spletno stran je prišel s klikom na povezavo na drugi spletni strani),"browser search" (na spletno stran je prišel preko iskalnika, ki je vgrajen v spletni brskalnik). Ta podatek lahko pove veliko o navadah uporabnika in tudi o pomembnosti spletne strani. Če ima stran veliko obiskov in do nje uporabnik dostopa preko iskalnika in ne priljubljenih, lahko pomeni, da sam ni dobro seznanjen s to dobrino spletnih brskalnikov. Pri strani, ki je med priljubljenimi, vendar je njeno število obiskov relativno majhno, bi lahko vtičnik opozoril uporabnika, da te spletne strani ni že dolgo obiskal in da jo lahko odstrani. S tem bi lahko pripomogli k boljši organiziranosti (vtičnik bi lahko sam kategoriziral zaznamke glede na vsebino in čas v katerem so bili dodani - uporabno pri projektnem razvoju), pa tudi sami relevantnosti zaznamkov (če bi za graditev uporabnikovega profila uporabljali tudi strani iz priljubljenih).

time čas v katerem je prišlo do vnosa v podatkovno bazo. Atribut je prisoten v vseh tabelah in je uporaben za povezovanje različnih vnosov med seboj. Ob vsakem obisku spletne strani pride do novega vnosa v vse tabele in jih samo s pomočjo identifikacijske številke strani ne moremo povezati. Možnost bi bila tudi generacija unikatnega identifikatorja za vsak obisk, vendar se mi je zdel čas veliko bolj uporaben - brez povezovanja tabel lahko vidimo, kdaj je prišlo do vnosa.

[2] `http://outfit7.com/`

6.3.4 Tabela "sizes"

Tabela vsebuje podatke o dimenzijah spletne strani, s katerimi se sreča uporabnik. Vse velikosti so predstavljene v pikslih.

pageid - identifikacijska številka spletne strani.

viewx in viewy - vrednosti predstavljata vidno polje znotraj okna spletnega brskalnika. Velikost tega območja je odvisna od velikosti zaslona, števila orodnih vrstic, ali imamo okno razširjeno skozi celotno velikost zaslona, itd. Dve vrednosti sta zato, ker predstavljata dolžini po x in y osi.

sizex in sizey - ti dve vrednosti predstavljata velikost celotnega HTML dokumenta (ko se celotna vsebina naloži in prevede v brskalniku).

top - za koliko je vidno območje spletne strani premaknjeno od zgornjega roba HTML dokumenta.

left - za koliko je vidno območje spletne strani premaknjeno od levega roba HTML dokumenta.

maxx in maxy - največje število pikslov, za katere se je miška premaknila od levega in zgornjega kota.

time - time ima enako vlogo kot v prejšnjem primeru, zato se bom opisu izognil. Enako velja za pageid, katerega bom v nadaljevanju navajal, vendar ga ne bom opisal.

Ti atributi so v povezavi med seboj lahko uporabni za različne zadeve. Pridobi se lahko delež strani, ki si ga je uporabnik ogledal. Pove tudi do katere točke se je spustil pri svojem brskanju. To je pomembno, saj je lahko izpustil določene informacije, ki se nahajajo na dnu strani in so zanj pomembne. Vtičnik bi ga v tem primeru lahko opozoril, da ga lahko novica, ki je na dnu strani, zanima. Ena izmed možnosti uporabe bi lahko bila premik elementov v primeru, da ima uporabnik prenosnik, ki je doma priklopljen na večji zaslon. Tekom dneva ima lahko sestanek, gre v službo, ipd. in se odklopi od večjega ekrana, kar pomeni, da ima sedaj veliko manj prostora za prikaz vseh podatkov. Če se spletne strani, ki jih uporabnik v tem času obiskuje ponavljajo, bi lahko vtičnik drugače razporedil elemente in jih tako prilagodil za manjši zaslon (oziroma bolj uporabljene elemente bi premaknil drugam). To je danes še posebej pomembno, saj se večina spletnih aplikacij mora prilagoditi za zaslone telefonov, tabličnih računalnikov, ipd. Da ne bi bilo potrebno razvijati različnih aplikacij, bi se samo vključilo programsko kodo, ki bi to avtomatsko prilagodila.

6.3.5 Tabela “coordinates”

pageid

time

x in y - predstavljata x in y koordinati miške na spletni strani v določenem trenutku. Premiki se shranjujejo vsako sekundo in s povezavo posameznih točk lahko prikažemo pot uporabnikove miške po spletni strani.

Zaznavanje premikov miške je lahko uporabno pri ugotavljanju mest, kamor je namenjena pozornost uporabnikov. V povezavi z drugimi podatki, bi lahko ugotovili tudi, kam uporabnik premakne miško, ko bere določeno besedilo.

Premiki in lokacije miške so še posebej uporabne pri osebah, ki niso preveč spretne z računalnikom. Glede na premike bi bila olajšanja tudi podpora za določen produkt, saj bi se takoj videlo, če je uporabnik sploh poskusil klikniti na določeno območje. Pri podpori je možna rešitev uporaba oddaljenega namizja (angl. “remote desktop”), vendar je tu potrebna privolitev uporabnika za dostop do oddaljenega namizja.

6.3.6 Tabela “advertisements”

V tabelo se shranjujejo podatki o oglasih, ki se nahajajo na spletni strani.

pageid

time

position - x in y koordinati (ločeni s presledkom) oglasa na spletni strani. Veliko raziskav je bilo že narejenih in je znano katera območja so na spletnih straneh bolj primerna za oglase. Nekateri razvijalci se odločajo za premikanje oglasov na nestandardna mesta. Tu je lahko veliko klikov, ki se zgodijo nenamenoma in je zaradi tega oškodovan oglaševalec. Pri tem je lastnik spletne strani zadovoljen, saj mu to pomeni dodaten vir zasluška (če gre za oglaševanje na klik), vendar je lahko za uporabnika moteče in ga lahko odvrne od obiskovanja strani. Vtičnik bi lahko lastnika strani opozoril, da je oglas na neprimernem mestu.

clicked ali je bil izveden klik na oglas ali ne. To nam pove, ali se uporabnik dejansko zanima za oglas ali ne. Če je kliknil pomeni, da ga vsebina oglasa zanima in glede na vsebino oglasa (oziroma spletno stran, ki se odpre ob kliku) dobimo nove informacije o uporabnikovih interesih.

Kontekstno oglaševanje je zelo pomemben del današnjega spleta. Prinaša prednosti za vse: lastniki spletnih strani in spletne oglaševalske mreže si delita zaslužek, oglaševalci pridobijo nove potrošnike in spletni obiskovalci pridobijo uporabne povezave. Relevantnost oglasov, ki se na strani prikazujejo je zato bistvenega pomena. Večina spletnih oglaševalskih mrež deluje še vedno po principu, da obdelajo vsebino spletne strani in prikazujejo oglase glede na pridobljene rezultate [5]. Nekatere oglaševalske mreže pa se le odločajo za nove pristope in s pomočjo umetne inteligence in uporabnikovega profila (ki ga pridobijo s podatki iz spletnega brskalnika) prikazujejo oglase, ki so najbolj relevantni za obiskovalca. Vtičnik bi lahko z zaznavanjem oglasov in uporabniškim profilom deloval kot posrednik in pomagal spletnim oglaševalcem, da pridobijo boljše informacije in tako bolje ciljajo oglase. Brez da bi oglaševalec poznal uporabnika, bi vtičnik samo poslal zahtevo, da se oglasi z določenimi vsebinami prikazujejo na spletni strani. Če vtičnik zazna oglase, jih lahko tudi skrije v primeru, da njihovo prikazovanje ni potrebno. Veliko spletnih uporabnikov ima tako jutranjo rutino (pregleda pošto, spletne strani z novicami, itd.). Pri tem jih oglasi ne zanimajo in se lahko, v povezavi z drugimi atributi, prikazovanje oglasov prepreči.

6.3.7 Tabela “elementclicks”

V to tabelo se shranjujejo vsi elementi, na katere uporabnik klikne.

pageid

time

element - oznaka elementa nad katerim je bil izvršen klik.

clicks - število klikov na ta element (kliki se štejejo za vsak obisk posebej).

posx in posy - položaj elementa na spletni strani, x in y koordinati.

S tem pridobimo število klikov na spletno stran in kje so bili ti kliki izvršeni (ne pridobimo točno lokacijo klika, ampak območje, v katerem je bil klik - lahko bi tudi pregledovali vse lokacije klikov in ustvarili t.i. vroče cone (angl. “hot zones”), torej mesta, kjer je prišlo do največjega števila klikov in to bi lahko tudi izrisali na spletni strani). Med drugim bi bilo to zelo uporabno v primeru, da bi vtičnik sam razporejal elemente spletne strani in bi tako na bolj vidna mesta postavil pomembnejše elemente.

6.3.8 Tabela "inputboxes"

Vsebuje nize, ki jih uporabnik vpiše v vnosna polja na spletni strani. Vsako vnosno polje prestavlja eno vrstico tabele.

pageid

time

boxid - oznaka (id) vnosnega polja v katerega je bil tekst vpisan.

words - vsebina, ki je bila vnesena v polje.

Tu lahko vtičnik zazna kaj si uporabnik dopisuje v spletni klepetalnici, piše v e-poštno sporočilo, itd. (če je to s strani spletne aplikacije dovoljeno in nima kakšnih varnostnih nastavitev proti takemu beleženju). Vsi tej podatki so v kasnejši obdelavi veliko bolj pomembni od vsebine spletne strani, saj so direktno povezani z uporabnikom. Z zaznavanjem takih vnosov bi lahko obogatili tudi rezultate samih spletnih iskalnikov, saj bi pri vpisu določenega niza poleg rezultatov, ki jih vrne npr. Google, dodali še druge rezultate, ki se zgenerirajo ob uporabi dodatnih podatkov iz uporabniškega profila (katerega je prej seveda potrebno realizirati).

6.3.9 Tabela "shortcuts"

V prejšnjem podpoglavju sem opisal shranjevanje nizov, ki jih uporabniki vnesejo v vnosna polja. Če pa ni fokusa v nobenem polju, pomeni, da je uporabnik po nesreči pritisnil kakšno tipko oziroma, da je uporabil bližnjico.

pageid

time

value - katera tipka je bila pritisnjena npr. CTRL + V.

timesused - koliko krat je bila bližnjica uporabljena. Za vsak obisk se ustvari nov zapis.

Spletne obiskovalce, ki ne uporabljajo bližnjic bi lahko seznanili, da obstajajo in s tem lahko pohitrili njihov spletno izkušnjo. Za ostale, ki pa bližnjice uporabljajo lahko sklepamo, da so bolj izkušeni uporabniki.

6.3.10 Tabela "previouseurl"

Zapis se ustvari v primeru, da je uporabnik prišel na spletno stran iz druge strani.

pageid

time

url - naslov spletne strani, iz katere je na stran prišel.

To je lahko uporabno, saj dobimo zaporedje strani, ki jih uporabnik obišče. Večina ljudi ima vsakodnevne rutine in bi lahko s pomočjo tega že vnaprej pripravili (shranili v predpomnilnik) spletne strani, katere bo uporabnik obiskal ob naslednjem kliku.

6.3.11 Tabela "time"

Ključni element brskanja po spletu je čas. Pozornost sem namenil naslednjim podatkom.

pageid

timeonpage - koliko časa je uporabnik imel spletno stran odprto.

dateoff - ob kateri uri je uporabnik zapustil spletno stran.

longestidle - najdaljši časovni interval, v katerem uporabnik ni izvedel nobene akcije na spletni strani (premik miške, pritisk tipke na tipkovnici, itd.). S tem lahko pridobimo podatek o uporabnikovi aktivnosti na spletni strani. To je zelo dober pokazatelj produktivnosti oseb in je veliko produktov (npr. Hamster[3]), s katerim se da to nadzorovati.

S povezavo časa, ki ga je uporabnik preživel na strani in časom, ko je stran zapustil lahko pridobimo časovni interval, v katerem je spletno stran obiskal. Če se ta čas ponavlja, se lahko stran že vnaprej pripravi v predpomnilniku, saj lahko pričakujemo, da bo sledil obisk.

S pomočjo časa bi se dejansko dalo uporabnikom "odpreti oči" in jim prikazati, koliko časa dnevno porabijo za neproduktivne stvari (Youtube, Facebook, itd.). Izdelalo bi se preprost graf, ki bi prikazal tudi, kdaj v dnevu pade efektivnost dela. Delodajalec bi lahko v tem času zaposlenim zadal naloge, ki zahtevajo manjši fizični in umski napor.

[3] `http://projecthamster.wordpress.com/`

6.3.12 Tabela “scrolling”

Brskanje po spletnih straneh, ne bi bilo tako enostavno brez srednje miškine tipke in hitrega pomikanja. Tabela shranjuje vrednosti, ki govorijo o temu, ali je do pomikanja prišlo ali ne.

pageid

time

yesorno - “yes”, v primeru, da je uporabnik pritisnil srednji gumb na miški. “no” je privzeta vrednost in pomeni, da ga ni pritisnil.

Če se uporabnik pomika s srednjo miškino tipko lahko predpostavimo, da si ogleduje vsebine spletne strani. Če to lastnost povežemo s časovnimi lastnostmi lahko ugotovimo ali uporabnik tudi bere vsebine spletne strani.

6.3.13 Tabela “links”

V tabelo se shrani celoten spisek povezav (angl. “link”) s spletne strani.

pageid

time

url - naslov spletne strani.

clicked - ali je prišlo do klika na povezavo ali ne. S tem lahko dobimo kako so si obiski spletnih strani sledili.

Zakaj bi nekdo sploh pričakoval da so linki uporabni? Hiperpovezava (angl. “hyperlink”) je referenca strani b, ki se nahaja na strani a. Ko kliknemo na povezavo v spletnem brskalniku se prikaže spletna stran b. Sama po sebi ta funkcionalnost ni kaj preveč uporabna pri pridobivanju informacij. Lahko pa način, kako so povezave uporabljene s strani avtorjev spletnih strani prinese veliko dodano vrednost. Običajno avtorji dodajo zunanje povezave, ker menijo, da bodo uporabne za obiskovalce spletne strani. Kljub temu, da je veliko število povezav takih, ki so namenjene samo pomoči uporabnikom (npr. ob kliku preusmeri uporabnika nazaj na začetno stran) se pojavljajo tudi take, ki sovpadajo z vsebino spletne strani. Druga vrsta povezav lahko uporabnika pripelje do zelo kvalitetnih spletnih strani, ki so močno povezane s tematiko strani, ki povezavo vsebuje. Analiza, ki jo opisuje članek [31] se naslanja ravno na zgornji tekst in temelji na naslednjih predpostavkah:

- povezava iz strani a do strani b je priporočilo strani b s strani avtorjev strani a.
- če sta stran a in b povezani s povezavo je verjetnost, da pišeta o isti tematiki večja, kot pa če nista povezani.

Torej lahko s pomočjo zgornjih predpostavk in profila uporabnika posodobimo npr. Googlovo poizvedbo s spletnimi stranmi, ki so za uporabnika bolj relevantne.

6.3.14 Tabela "vocabulary"

Na spletu so razpoložljive ogromne količine teksta in ročno beleženje bi bilo skrajno neučinkovito in počasno, zato je potrebno uporabljati različne tehnike za avtomatsko beleženje.

Tabela predstavlja spisek vseh besed, ki jih vtičnik na spletni strani beleži.

pageid

beseda - nizi, ki jih pošlje vtičnik se z regularnim izrazom ločijo na posamezne besede.

tipbesede - Google in veliko ostalih aplikacij sestavi podatkovno strukturo iz celotne vsebine spletne strani. To pa je prevelika količina podatkov in velik del slednjih za uporabnika nepomembnih, zato sem se v prvi verziji vtičnika odločil za shranjevanje naslednjih besed:

- vse besede, ki predstavljajo metapodatke (saj naj bi bile te že same po sebi dobro premišljene in povedo veliko o sami vsebini spletne strani).
- tekst, ki ga uporabnik počrni. Tu predpostavljam, da je ta tekst za uporabnika veliko bolj pomemben kot druge vsebine.
- tekst, ki ga uporabnik kopira (kombinacija tipk CTRL-C).

teza - vsak tip besede ima svojo težo, saj so nekateri tipi besed bolj pomembni od drugih. Posamezne uteži sem določil po lastni presoji, brez upoštevanja predhodnih raziskav.

stpojavitev - vsaka beseda se lahko znotraj konteksta pojavi večkrat. Da se nebi za vsako vnašal nov zapis, se enake besede sešteje in število pojavitev vpiše v to polje.

6.3.15 Tabela "keywords"

V tabeli so predstavljene ključne besede za vsako spletno stran. Vsaki besedi se določi relevantnost, katero se izračuna glede na število pojavitev in določeno utež. Vira besed sta tabeli "vocabulary" in "inputboxes".

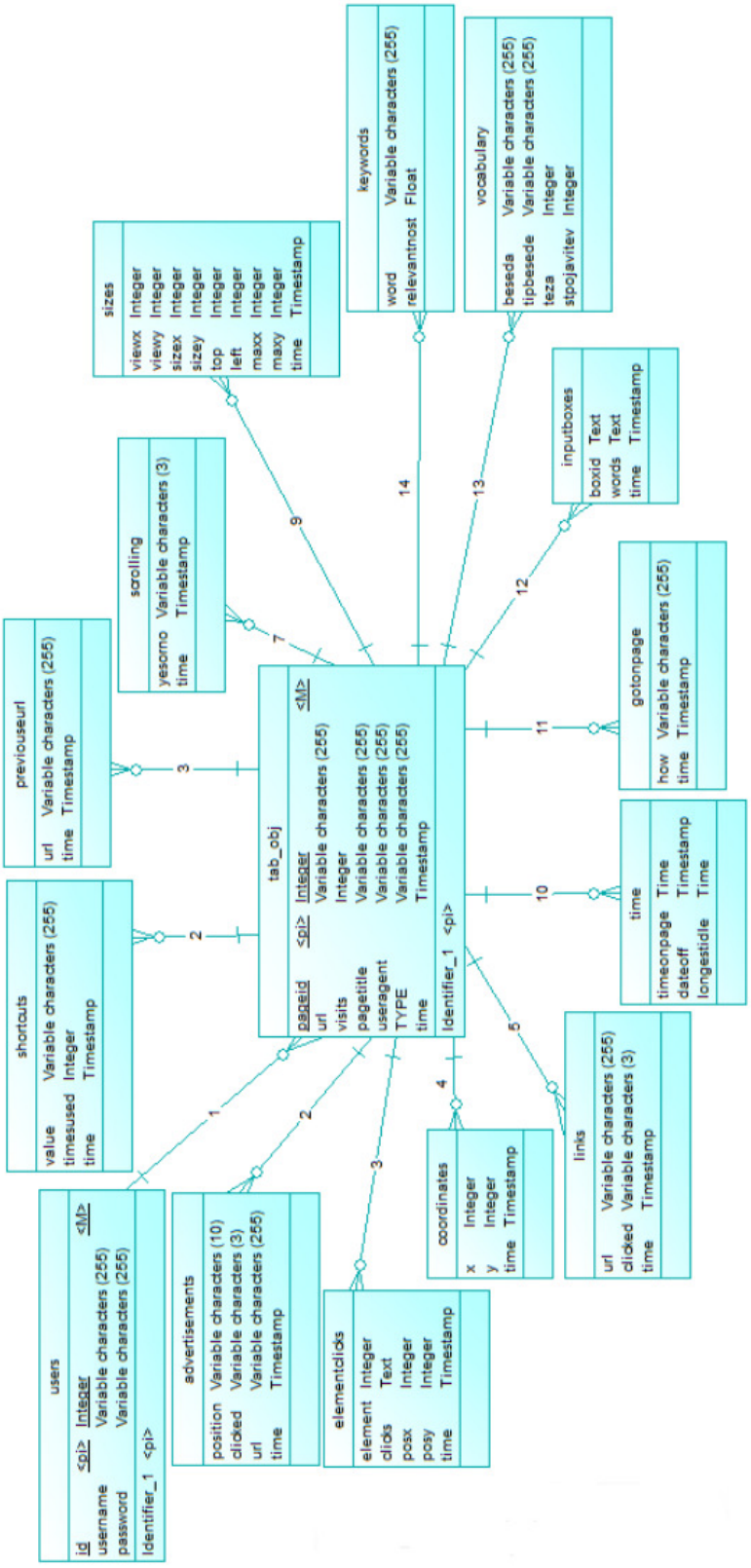
pageid

word - ključna beseda.

relevantnost - izračun relevantnosti je zelo preprost. Za vsako besedo (glede na tip) se zmnoži število pojavitev in utež ter na koncu sešteje vrednosti za iste besede različnih tipov.

Ključne besede se izpišejo v pojavnem oknu in na hitro obnovijo vsebino spletne strani za uporabnika. Slednji bi lahko to uporabil pri iskanju sorodnih vsebin, saj bi mu vtičnik lahko predlagal besede, na katere sam nebi pomislil.

Pri ključnih besedah je prisotna še ena velika pomanjkljivost, saj se na vrh seznama relevantnosti pojavljajo besede kot so: ko, na, in, ipd. Potrebno bi bilo izdelati slovar, v katerem so te besede prisotne, da se jih ne bi upoštevalo pri izpisovanju.



Slika 6.2: Konceptualni diagram podatkovne baze.

Poglavje 7

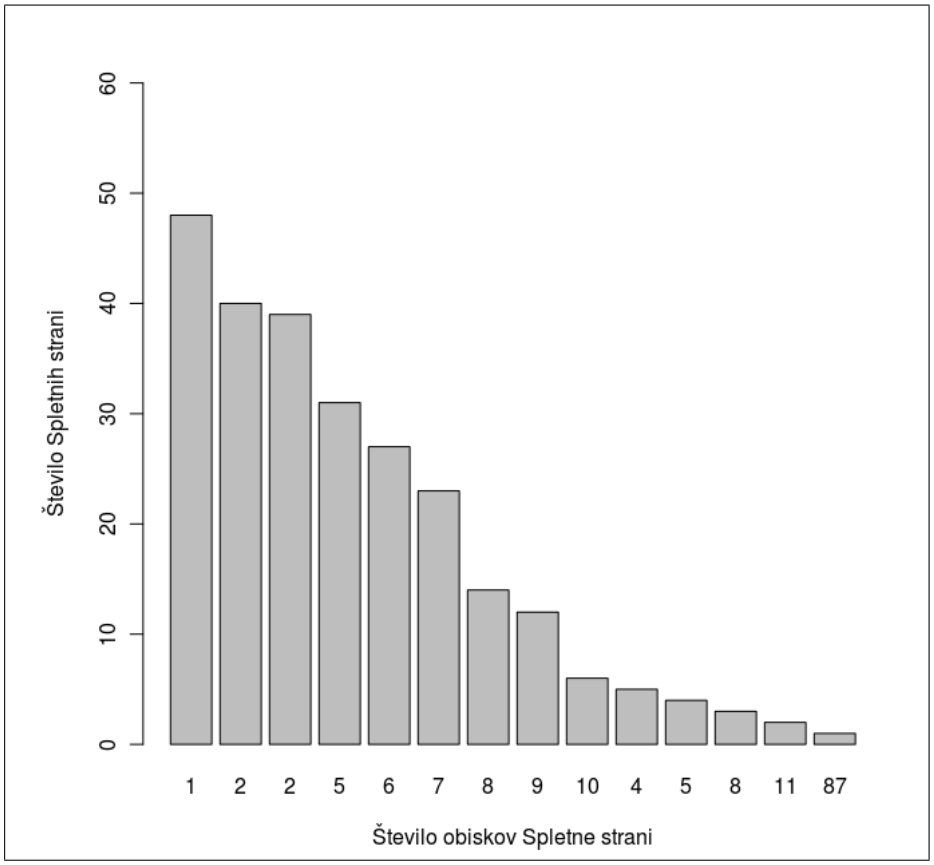
Rezultati

V testiranje delovanja vtičnika je bilo vključenih 6 uporabnikov. Trije študenti, mlajši od 25 let (dva ob študiju delata kot programerja, en pa kot špediter), dva sta med 25 in 30 že zaposlena in en uporabnik je starejši od 50 let in tudi zaposlen. Vseh šest uporabnikov je pri beleženju podatkov sodelovalo 5 delovnih dni. Med njimi sta dva imela vtičnik nameščen samo na delovnem računalniku, ostali pa tudi na domačem. Podatke sem za ostale štiri beležil tudi med vikendom. Ker je v vtičnik realizirano samo beleženje podatkov, sem za boljšo analizo in predstavitev rezultate tudi ročno pregledal in tako dobil bolj podrobne informacije. Skupno je bilo obiskanih 124 različnih spletnih strani (skupno 496 obiskov). Najbolj obiskana spletna stran je bila 24ur.com, kar priča o temu, da kljub pritoževanju Slovencev, da so njene vsebine samo prevodi drugih novic, ostaja resno najbolj priljubljena. Sledijo ji Finance.si, Facebook in Gmail. Na sliki 7.1 lahko bolj podrobno pregledamo statistiko obiskov.

Z uporabo vtičnika sem poskusil pridobiti vzorce vedenja uporabnikov, ki so unikatni in značilni za posameznika. Za potrditev pridobljenih podatkov sem brskanje uporabnikov za nekaj minut tudi opazoval. S tem sem poleg potrditve že znanih vzorcev pridobil tudi značilnosti in informacije, ki so bile v zabeleženih podatkih skrite in bi jih pri analizi težko opazil.

7.1 Analiza

Pri analizi se bom osredotočil na vse, kar sem z vtičnikom beležil in opisal pridobljene rezultate. Podatki se pri uporabnikih za katere sem beležil tudi vikend, močno razlikujejo. Spletna aktivnost med vikendom je manjša kot med delovnim tednom, razlikujejo pa se tudi vsebine. Že samo ta informacija



Slika 7.1: Število obiskov spletnih strani v razmerju s številom spletnih strani

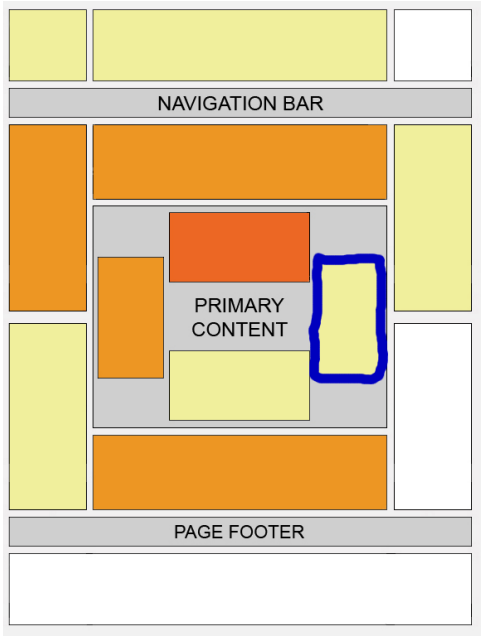
lahko spremeni načrt razvoja aplikacije, ki bi te podatke obdelovala. Sestavilo bi se lahko dva različna uporabniška profila (lahko tudi več), ki bi bili povezani z različnimi področji uporabnikovega življenja. To bi lahko bila nadgradnja kategorij in uporabnik bi lahko izbiral kateri uporabniški profil bi ustrezal trenutnemu brskanju. Profili bi se med seboj tudi prepletali, da bi bila zadeva še bolj učinkovita.

Aktivnost na spletu je med vikendom padla za nekaj več kot 75 % (iz 1399 minut na 335 minut). Med četverico uporabnikov eden brskalnika sploh ni uporabljal. Podobni podatki kot za vikend, veljajo tudi za "popoldansko" brskanje med delovnim tednom.

Sledi opis posameznih pridobljenih informacij in analiza.

Oglasi

Zaznanih je bilo 546 oglasnih polj v katerih so uporabniki izvršili 3 klike. To predstavlja 0.5 %, kar je za spletno oglaševanje kar dobro razmerje (razmerje med prikazi in kliki je dobro, če je nad 0.3 %), vendar je vredno pogledati, kaj se je po klikih zgodilo. Uporabnik, ki je klik izvedel, se na spletni strani, ki je sledila ni zadržal več kot nekaj sekund, kar pomeni, da je prišlo do klika pomotoma, ali pa ni bil zadovoljen s prikazano vsebino. Tudi oglasno območje v katerem so bili kliki izvršeni je bilo vedno enako. Spodnjo sliko 7.2 so izdelali pri Googlu in prikazuje mesta, kjer oglasi dosežejo največjo uspešnost. Na podobna mesta so bila umeščena oglasna polja tudi na spletnih straneh, ki jih je vtičnik zabeležil (velikosti posameznih polj se ne beleži, ampak le pozicija - v naslednji različici vtičnika bi bilo potrebno dodati tudi ta podatek). Mesto, nad katerim so bili izvršeni zgoraj omenjeni kliki, je obkroženo z modro barvo.



Slika 7.2: Donosnost mest gre od temno oranžne do bele barve. Z modro barvo je obkroženo mesto, kjer so bili izvršeni kliki (vir: [15]).

Velikokrat pa se zgodi, da se na desni strani ob glavni vsebini spletne strani namesto oglasov pojavi vsebina (ali obratno). Za nekatere uporabnike je to lahko moteče, saj se privadijo, da je tam npr. meni z

najnovejšimi novicami. Na drugih spletnih mestih pričakujejo enak razpored in jih to zavede da kliknejo na oglas. To se še posebej pozna pri oglasih, ki se prikažejo glede na vsebino spletne strani. Zanimiv je tudi podatek, da skoraj vse strani vsebujejo vsaj eno oglasno polje.

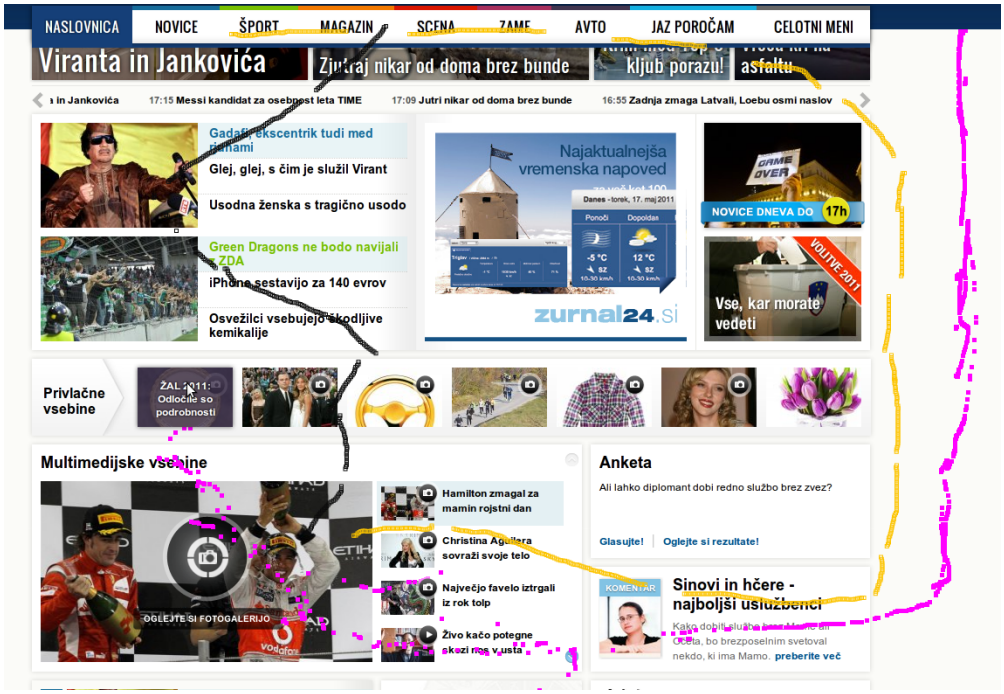
Premiki miške in pomiki s srednjim miškinim gumbom

Za vse uporabnike je bilo značilno vedenje, da so si pri brskanju po spletnih straneh pomagali s koleščkom na miški. Zanimivo bi bilo pogledati, kako bi se današnji spletni uporabniki znašli in koliko več časa bi porabili, če bi imeli na razpolago starejše tipe miške, brez koleščka.

Medtem, ko je uporabnik držal pritisnjen kolešček, se je miška v večini primerov nahajala na desni strani zaslona. Tu si je najverjetneje ogledoval vsebine spletne stran in pomaknil miško na to mesto, da ga ne bi motila. Značilnost je bilo možno opaziti pri vseh uporabnikih. V manjšem številu primerov, se je kurzor nahajal bolj proti sredini stran, torej nad glavno vsebino (v primeru standardne organizacije elementov spletne strani).

Sami premiki miške po strani prikazujejo dejstvo, da so uporabniki že izkušeni, saj ne izvajajo nepotrebnih premikov. Tu bi bilo za testiranje zelo dobro vključiti kakšen spletni kviz, ki zahteva od uporabnika, da rešitve najde na sami spletni strani. Tako bi lahko primerjali in videli različne rezultate. Slika 7.3 prikazuje primer premikanja miške treh uporabnikov na spletni strani z novicami.

Zanimivi so tudi vzorci vedenja med branjem določenih vsebin. Da se je uporabnik odločil za branje sem sklepal v povezavi s časom na strani, časom v katerem se miška ni premikala in pomikanjem po spletnih straneh. Večinoma so uporabniki pri branju vsebine miškin kurzor premaknili na stran, da jih ta pri branju ne bi motila. V nekaterih primerih, se je kurzor iz statičnega mesta premaknil za nekaj pikslov, kar bi lahko pomenilo, da je uporabnik hotel poudariti določen del vsebine. Pri enemu uporabniku se je pojavil tudi vzorec kroga. Tu je uporabnik želel poudariti določeno vsebino s kroženjem 7.4 kurzorja miške okoli njega. Tu bi bilo zelo pomembno preveriti ali je to uporabil za lastno informiranje ali za razlago drugi osebi. Ta podatek bi lahko na spletnih straneh (ali tudi sam vtičnik) uporabili, da bi pritegnili pozornost uporabnika.



Slika 7.3: Različne barve prikazujejo premike miške različnih uporabnikov, da bi dosegli določeno novico. Vtičnik zaznava te podatke, vendar sem ta poizkus naredil na istem računalniku v nekaj minutnem zamiku.

Povezave

Klikov na povezave je bilo največ na spletnih straneh z novicami. Te povezave so v večini primerov vodile na podstrani z novicami. Povezave, ki so vodile na zunanje strani (in jih je uporabnik kliknil) so bile (glede na čas, ki ga je uporabnik prebil na naslednji strani) kar učinkovite. V povprečju je uporabnik na teh straneh brskal 4 minute.

Čas

Skupaj je bilo 11130 minut beleženja podatkov. Razporeditev med uporabniki je prikazana na sliki 7.5.

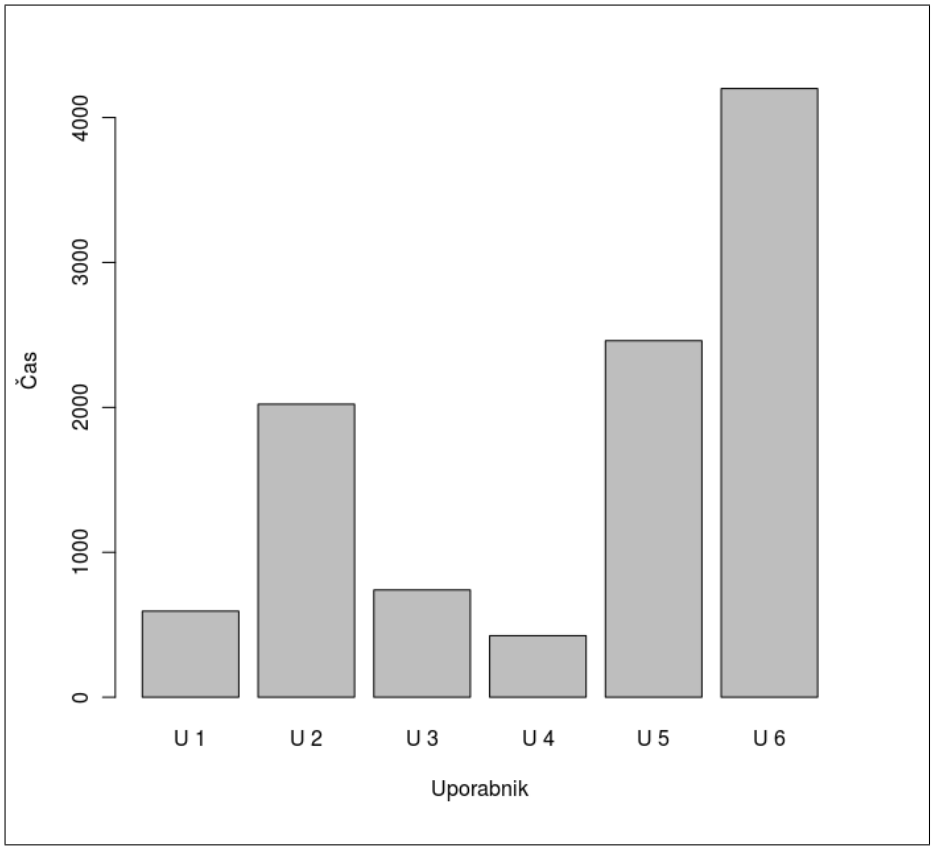
Najbolj priljubljene spletne (tiste z velikim obiskom) strani lahko povedo veliko o delovnih navadah uporabnika. Za trenutek se osredotočimo samo na te spletne strani in na sliki 7.6 poglejmo, v katerih časovnih intervalih so jih uporabniki obiskovali. V večini gre to za strani, ki nimajo ve-

Slika 7.4: Primer kroženja miške okoli določene vsebine, ki jo je uporabnik želel povdariti.

liko skupnega z zaposlitvijo uporabnika, oziroma jim ne pomagajo pri povečanju produktivnosti, zato je pričakovano, da jih bodo obiskali ob prihodu na delovno mesto ali pa v času kosila. Obiski teh strani so se beležili tudi vmes, vendar v nobenem posebnem zaporedju. Veliko je tu odvisno od dnevne količine dela in zmožnosti koncentracije. Uporabnik 2 si vsakih 45-60 minut vzame nekaj minut za brskanje po straneh, ki niso povezane z njegovim delom (programiranjem).

Na posamezni spletni strani so se uporabniki zadrževali od nekaj sekund do več ur. Na nekaterih straneh, ki so bile odprte več časa se je pojavil tudi velik čas (angl. "idle time"), v katerem se na strani ni zgodila nobena akcija. To so bile strani, ki se sproti posodabljajo (email) ali pa se uporabljajo pri splošnih opravilih (Googlov prevajalnik[1]). Rekord je

[1] `http://translate.google.com`



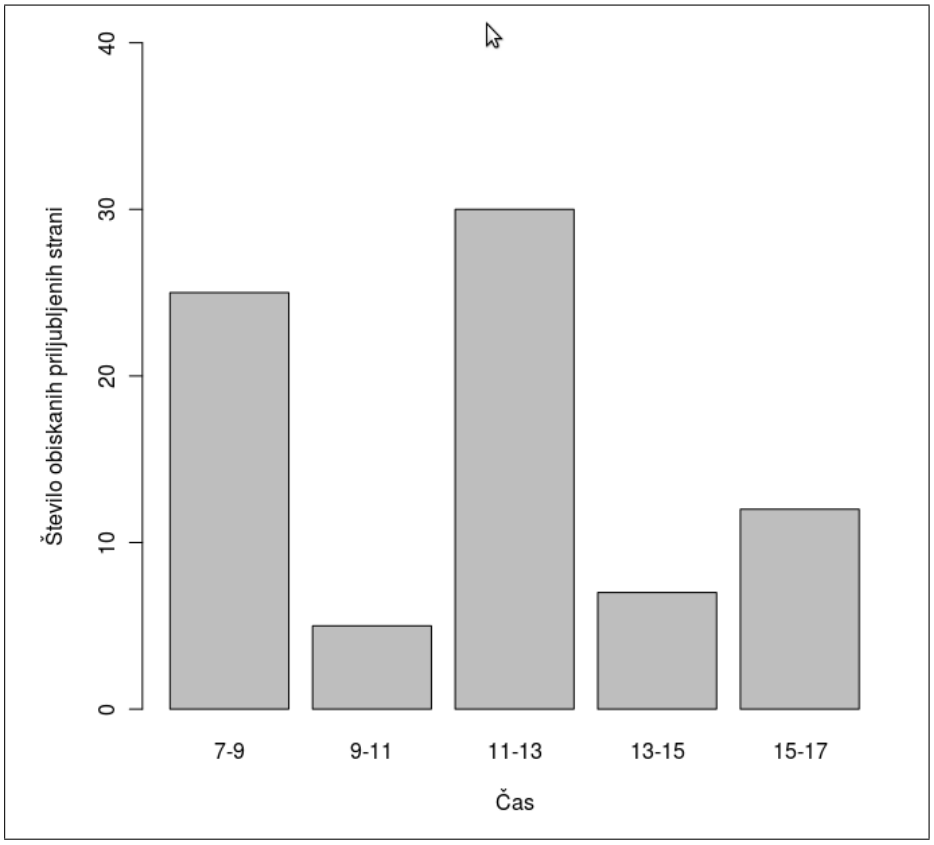
Slika 7.5: Uporaba vtičnika v minutah. Uporabniki 1,4,5 in 6 imajo dodan še čas dveh dodatnih dni (vikend).

dosegla stran, ki je bila odprta 8 ur in 14 minut (Gmail[2]).

Lahko bi se zgodilo, da bi uporabnik pustil za dalj časa odprto stran, ki vsebuje razne multimedijske vsebine. Če ne bi bil pozoren bi lahko taka stran hitro pripeljala do velike porabe sistemskih virov. Vtičnik bi ga lahko s kakšnim zvočnim signalom ali pojavnim oknom na to opozoril in preprečil potencialno nevarnost.

Pri procesu iskanja določene informacije so se pokazali zelo zanimivi podatki. Da je prišlo do iskanja sem določil s pomočjo zaporedja obiskanih spletnih strani in časa, ki ga je uporabnik porabil na posamezni strani. Na začetku tega zaporedja je bil v vseh primerih Googlov iskalnik[3]. Vsem

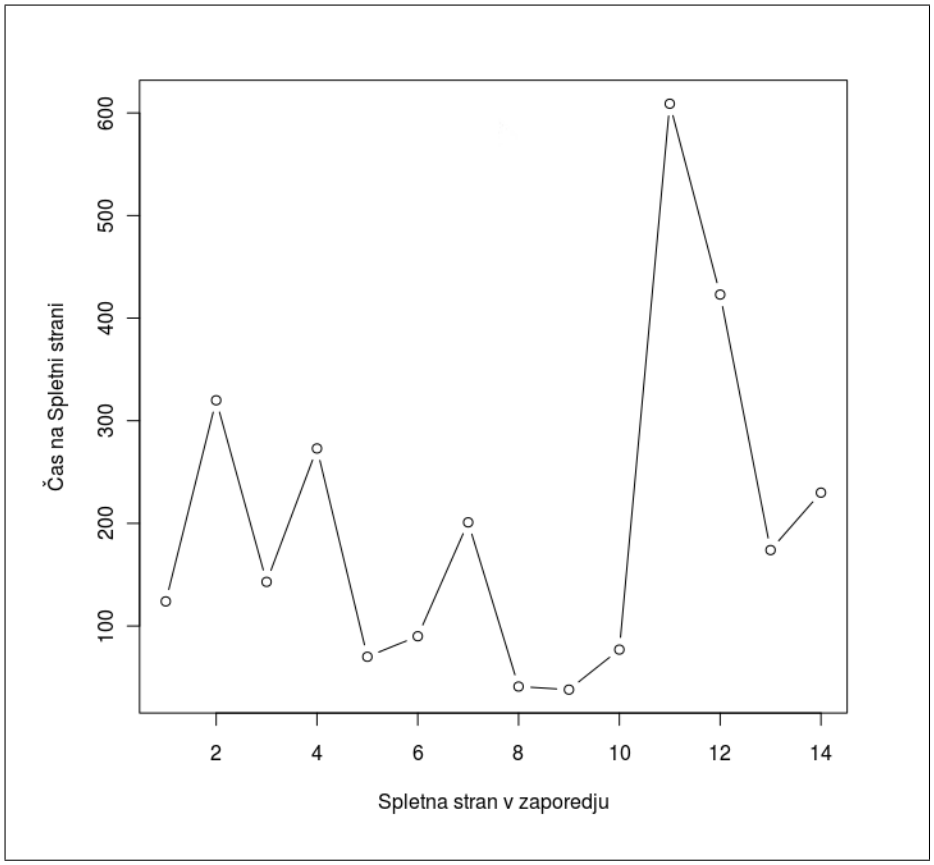
[2] `http://www.mail.google.com/mail?hl=sl`
[3] `www.google.com`



Slika 7.6: Kdaj tekom dneva so uporabniki obiskovali priljubljene spletne strani.

uporabnikom je skupno, da ne poznajo ukazov za pospešitev iskanja, ki jih v svojem iskalniku ponuja Google. Primeri so dostopni na [21]. Na splošno je zelo malo uporabnikov seznanjeno s tem in bi lahko v Google Chromu v primeru npr. obiska spletne stran s kalkulatorjem uporabnika opozorili, da se lahko preproste matematične izraze vpiše kar v iskalnik. Funkcijo bi lahko dodal tudi v vtičnik, vendar bi se rajši izognil privilegiranju enega brskalnika pred drugimi. Slika 7.7 prikazuje najdaljši postopek iskanja, ki ga je vtičnik zabeležil in čase na posameznih spletnih straneh. Tekom iskanja je uporabnik pregledal 14 strani in se medtem 3 krat tudi vrnil na domačo stran Googla in spremenil iskani niz.

Pri omenjenem zaporedju sem šel po njegovem postopek iskanja in pregledal, kaj ga je zadržalo na spletni strani (ukvarja se z programiranjem, tako, da mi zadeva ni bila tuja). Šlo je za iskanje razlage pri delu pro-



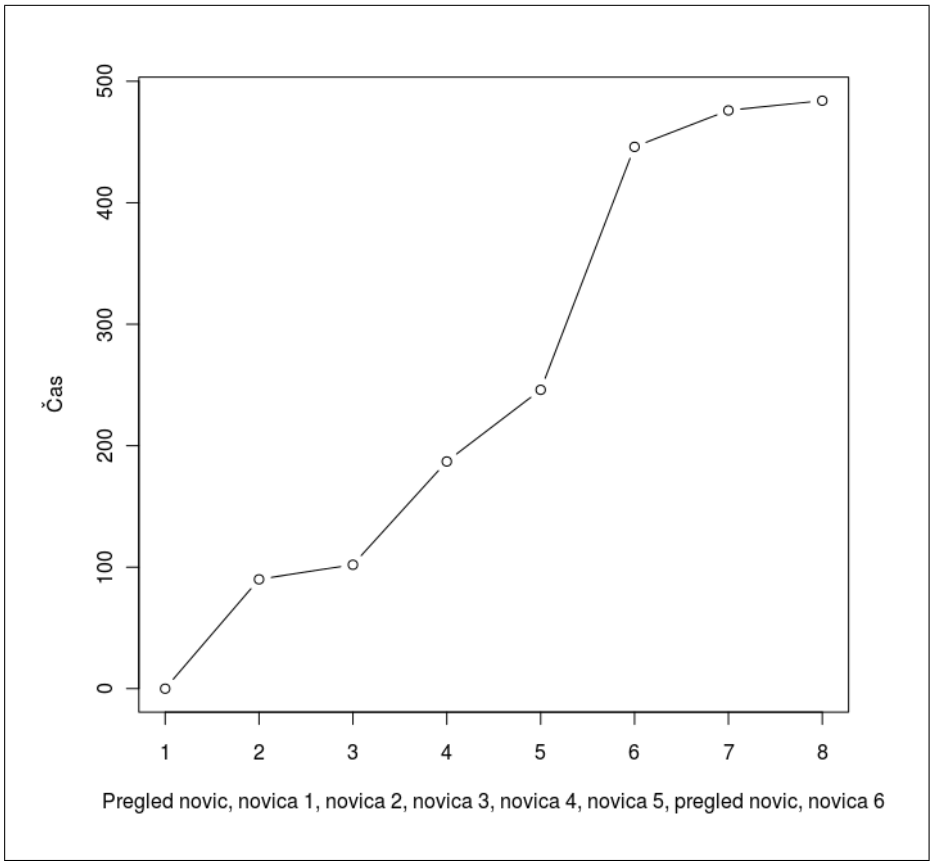
Slika 7.7: Zaporedje uporabnika pri iskanju določene informacije.

gramske kode. Kljub temu, da so nekatere druge strani vsebovale podatke, ki bi mu lahko prikrajšale iskanje se je dalj časa zadrževal na straneh, ki so vsebovale dele programske kode. Tu bi lahko vtičnik sam pripomogel k izključitvi strani, ki ne vsebujejo kode. V takih primerih bi pa bilo zelo težko prepoznavati različne vrste poizvedb.

Ostaja še čas, v katerem se miška ni premaknila. Enako kot za strani, se tudi pri novicah opazi, katere uporabnika bolj zanimajo. Le pri redkih novicah je bil čas dovolj dolg, da bi uporabnik lahko prebral celoten tekst (hitrost branja uporabnika lahko samo predpostavljam, če bi se podatki beležili več časa, bi se ta podatek pridobilo s statističnim izračunom). Pri ostalih je verjetno prebral le uvodni del oziroma samo preletel novico. Slika 7.8 prikazuje uporabnikov obisk spletne strani Žurnal 24[4]. Celoten

[4] `http://www.zurnal24.si/`

čas na spletni strani je trajal 484 sekund (približno 8 minut). Najprej je bil uporabnik na osnovni strani in se pri tem pomikal po strani navzdol, kar najverjetneje pomeni, da si je ogledoval naslove novic. Pri tem je zanimive naslove odpiral v novih zavihkih (domača stran je bila še vedno odprta in ni bilo zaznanih premikov) in jih postopoma zapiral. Glede na ostale novice je občutno daljši čas pri novici, ki je na sliki prikazana pod številko 6. Preveril sem, koliko znakov je bilo pri posamezni novici in čas, ki ga je uporabnik preživel na strani, ter ugotovil, da se razmerje med številom besed in časom razlikuje od ostalih primerov. Miška se je pri tem premikala in je uporabnik pregledal tudi spodnji del strani, kjer se nahajajo komentarji. Tu ga ni zanimala samo novica, ampak tudi mnenja ostalih. Vtičnik bi mu lahko v tem primeru ponudil druge spletne strani, na katerih poteka pogovor o izbrani tematiki.



Slika 7.8: Zaporedje uporabnika pri branju novic na spletni strani.

Prikazane velikosti

Pri prikazanih velikostih gre predvsem za to, ali so si uporabniki ogledali strani v celoti. Ponavljal se je vzorec, da so se v oknu brskalnika v celoti prikazovale le strani, ki so jih na novo obiskali. Večino priljubljenih naslovov niso pregledali do konca, kar pomeni, da so na dnu stvari, ki zanje niso zanimive. To je pomemben podatek za lastnike spletnih strani, ki svojim stalnim obiskovalcem ne smejo prikazovati pomembnih vsebin na mestih, ki jih nikoli ne obiščejo. Če se vsebine v določenem oknu prilagajajo glede na določenega uporabnika (glede na pretekle izbire se prikazujejo predlogi za nove), zakaj se tudi razporeditev elementov ne bi prilagajala. To zahteva nekoliko več dela, vendar bi se lahko izkazalo za učinkovito.

Leksikon in uporaba hitrih tipk

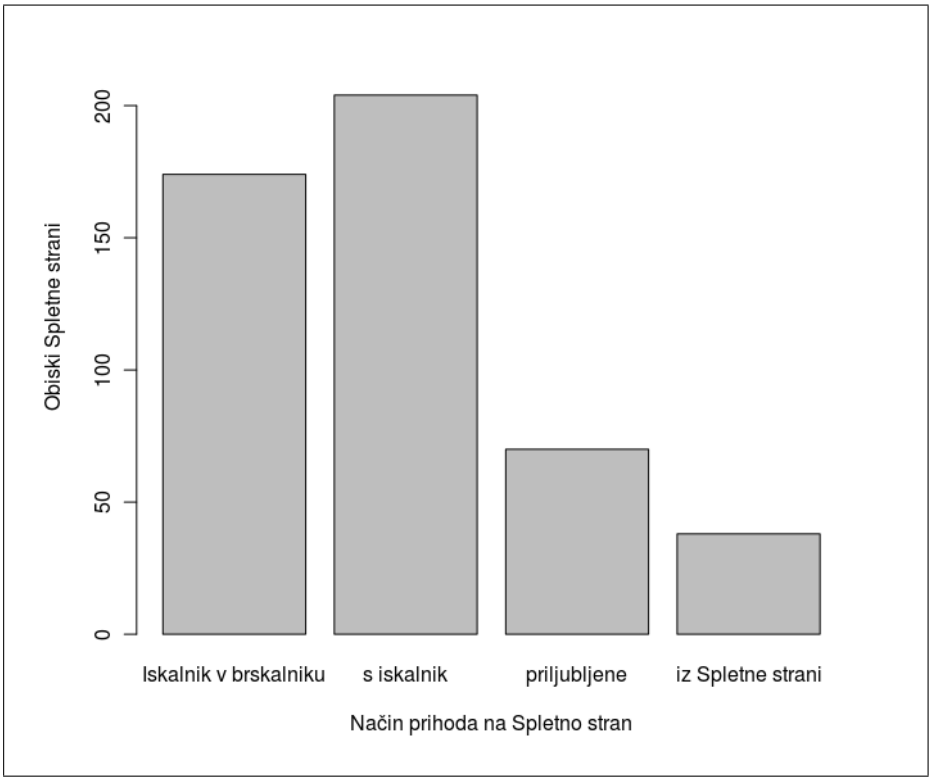
Pri brskanju je nekajkrat prišlo tudi do označevanja teksta. Večje količine teksta je vtičnik zabeležil le v redkih primerih in to le v procesu iskanja določenih informacij. Večkrat je prišlo do označevanja posameznih besed (povprečno enkrat na deset strani). Pri teh krajših nizih je prišlo tudi do uporabe hitrih tipk in predvsem standardnih kombinacij "CTRL+C" in "CTRL+V". Po kopiranju so bili nizi največkrat kopirani v vnosna polja (predvsem Googlov iskalnik). Poleg že omenjenih hitrih tipk, je bila zabeležena samo še kombinacija "CTRL-A". Vmes so se pojavile tudi posamezne črke, vendar so bile to verjetno napake in ne nekaj kar bi uporabnik resnično želel pritisniti.

V leksikon izrazov se shranjujejo tudi metapodatki strani. Nekoliko sem bil zaskrbljen nad kvaliteto metapodatkov, vendar se je izkazalo, da so kar dobro opisovali vsebino spletnih strani.

Način prihoda na spletno stran

Slika 7.9 prikazuje načine prihoda uporabnikov na spletne strani. Pomembno je poudariti, da imata dva uporabnika za privzet brskalnik Mozillo Firefox in sta Chrome uporabljala le za testiranje vtičnika in nista imela nastavljenih vseh bližnjic do priljubljenih strani.

Zanimiv je podatek, da kljub velikemu obisku nekaterih strani, jih uporabniki nimajo med priljubljenimi v brskalniku. So pa seznanjeni z obstojem priljubljenih in jih uporabljajo, saj imajo nekatere strani dodane.



Slika 7.9: Podatki o načinih prihoda uporabnikov na spletne strani.

7.1.1 Primerjava uporabnikov

Raznolikost uporabnikov je velika že v poklicu, ki ga opravljajo, zato se tudi njihove značilnosti brskanja med seboj razlikujejo.

Se pa hitro razbere dve različni skupini: ena predstavlja uporabnike, ki v svojem poslu uporabljajo programsko orodje, katero ima vse potrebne elemente vgrajene in tiste, ki si morajo pri iskanju rešitev pomagati s spletom. Tu lahko ločimo 2 programerja od ostalih 4ih, saj sta skupaj obiskala 72 od 87 strani, ki so bile obiskane samo enkrat. Če bi vnaprej poznali ta podatek (kdo uporablja brskalnik za zabavo in kdo za raziskovanje) bi se lahko že ob zagonu izbralo profil, ki bi bil optimiziran za uporabnika. Zelo zanimivo bi bilo poizkus ponoviti z osebo, ki uporablja spletni brskalnik za raziskovanje in ni povezana z računalništvom. Med skupinama, ki sem ju predstavil se razlikujejo tudi časi obiska posameznih spletnih strani. Pri programerjih so občutno krajši.

Obdobje testiranja je bilo vsekakor prekratko za ustrezno razdelitev uporabnikov v skupine. Tudi število uporabnikov je premajhno in raznolikost med

njimi prevelika, da bi poizkali vzorce vedenja, ki se medsebojno križajo.

Poglavje 8

Zaključek in nadaljnje delo

Beleženje podatkov lahko veliko pripomore k izboljšanju uporabniške izkušnje na spletu. Ustrezno razumevanje konteksta in njegova povezava s prilagodljivostjo je bistvena, če želimo zgraditi učinkovite aplikacije.

V delu sem se najprej poglobil v pomen besede kontekst in njeni povezavi s spletnimi uporabniki. V nadaljevanju poglavja sem predstavil še nekatere druge pojme, ki nastopajo v nalogi. Da bi dobil ustrezen pregled trenutnega stanja na tem področju, sem v 3. poglavju pregledal obstoječe rešitve in se nato odločil za vrsto aplikacije, ki jo bom razvil. Iz različnih razlogov sem se odločil za razvoj vtičnika, katerega bi uporabniki namestili in s tem beležili kontekst na obiskanih spletnih straneh. Dejstvo je, da je pri npr. Mozilli na voljo 2,5 milijonov vtičnikov ter da te razširitve koristi kar 85 % uporabnikov brskalnika [14].

Vsi spletni brskalniki omogočajo razvoj vtičnikov, zato sem v nadaljevanju naredil primerjavo med najbolj priljubljenimi (Opera, Internet Explorer, Safari, Google Chrome, Mozilla Firefox) in izbral Chrome. Izbira se je izkazala za dobro, saj razvoj temelji na popularnih spletnih tehnologijah (JavaScript, JSON in HTML) in imajo tudi odlično podporo za razvijalce. Težave sem imel le s shranjevanjem podatkov, saj se vtičnik poganja na računalniku odjemalca in je nekoliko problematično pošiljanje podatkov na strežnik.

Del naloge je bila tudi izbira in utemeljitev podatkov, katere bo vtičnik beležil. Nad izbiro sem zadovoljen, vendar bi bilo potrebno bazo še nekoliko razširiti. Funkcionalnost ključnih besed se je izkazala za neučinkovito, saj so bile besede na vrhu seznama večinoma nerelevantne za uporabnika in spletno stran. Ostali parametri so prikazali zanimive vzorce in informacije o uporabnikih, bi pa vsekakor potreboval večje število uporabnikov in daljše obdobje beleženja, za generiranje kakovostne analize.

Podatke sem ročno analiziral in predstavil nekatere primere in vzorce vedenja. Raznolikost izbranih uporabnikov za testiranje se ni izkazala za najboljšo. Glede na rezultate bi bilo boljše, če bi uporabil testno množico s podobnimi interesi in navadami. Na ta način bi lahko primerjal med seboj različne vzorce na istih straneh in bi s tem dobil informacije, ki bi bile relevantne tudi za lastnike spletnih strani.

Nadaljnje delo in razvoj vtičnika se mora v prvi fazi osredotočiti na izboljšanje načina shranjevanja podatkov. Trenutna različica se ne obnese najbolje pri velikih količinah zabeleženih podatkov. Če ostanemo pri podatkih, je potrebno urediti tudi lokalno shranjevanje v času brskanja po spletni strani. Sedaj se informacije shranjujejo v računalniški pomnilnik. Težavo bi se lahko rešilo z uporabo peskovnika (angl. "sandbox") s katerim bi se del podatkov lahko shranilo v navidezni prostor na računalniku odjemalca.

Naslednja različica vtičnika bi pa vsekakor morala preiti na višjo raven in omogočati tudi obdelavo pridobljenih podatkov.

Dodatek A

Namestitev in osnovna uporaba vtičnika

Za uporabo vtičnika je potrebno najprej namestiti spletni brskalnik Google Chrome. Po uspešni namestitvi lahko vtičnik pridobite tako, da pošljete e-pošto na naslov marko.jurincic@gmail.com.

Vtičnik se v brskalnik namesti tako, da klikne na gumb v obliki ključa na zgornji desni strani brskalnika, ter izbere možnost orodja in nato "Razširitve". Če možnost ni obkljukana je potrebno obkljukati okence in omogočiti možnost načina za razvijalce.

Vtičnik se naloži tako, da se klikne tipko "naloži odpakirano razširitev" in izbere mapo v kateri so shranjene datoteke vtičnika.

Po namestitvi se bo odprlo novo okno A.1, ki bo zahtevalo, da se uporabnik prijavi (oziroma registrira, če še nima uporabniškega imena in gesla). Da se ni potrebno prijavljati ob vsaki uporabi brskalnika lahko uporabnik izbere možnost "remember me", katere bo ustvarila piškotek, ki bo uporabnika prijavilo za obdobje enega leta.

Po uspešni prijavi v sistem se podatki na obiskanih spletnih straneh pričnejo beležiti in pošiljati na spletni strežnik, kjer se shranijo v podatkovno bazo. Če želi uporabnik preprečiti zaznavanje podatkovnega tipa to lahko stori s klikom na vtičnikovo ikono, ki se pojavi v zgornjem desnem kotu brskalnika in v na novo odprtem oknu A.2 odstrani kljukico pri izbrani možnosti.

Slika A.1: Prijava v vtičnik.

Slika A.2: Okno, ki se prikaže ob kliku na vtičnikovo ikono v spletnem brskalniku.

Slike

3.1 Piggy Bank - shranjevanje 12
3.2 Piggy Bank - pregled 13
3.3 The Lemur Project - orodna vrstica 14
3.4 The Lemur Project - uporabniški vmesnik 15
3.5 SemWeb - meni 16
3.6 SemWeb - nova stran s povezavami 17
3.7 Tržni delež brskalnikov 21

5.1 Primerjava porabe pomnilnika brskalnikov 25
5.2 Primerjava porabe pomnilnika brskalnikov - en zavihek 26
5.3 Primerjava porabe pomnilnika brskalnikov - 40 zavihkov 27
5.4 Primerjava porabe pomnilnika brskalnikov - vračanje virov . . . 28
5.5 Primerjava porabe pomnilnika brskalnikov - en zavihek 29
5.6 Primerjava porabe pomnilnika brskalnikov - 40 zavihkov 30
5.7 Primerjava porabe pomnilnika brskalnikov - vračanje virov . . . 31

6.1 Ikona vtičnika 39
6.2 Konceptualni diagram podatkovne baze 52

7.1 Obiskanost spletnih strani 54
7.2 Donosna mesta za prikaz oglasov 55
7.3 Primer premikanja miške 57
7.4 Primer kroženja miške 58
7.5 Uporaba vtičnika 59
7.6 Obisk priljubljenih spletnih strani 60
7.7 Zaporedje pri iskanju 61
7.8 Zaporedje pri pregledu novic 62
7.9 Prihodi na Spletno stran 64

A.1 Prijava v vtičnik 69

A.2 Prikazno okno 69

Literatura

[1] Ahmet Soylu, Patrick De Causmaecker, Piet Desmet, "Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering", v zborniku *JOURNAL OF SOFTWARE*, ACADEMY PUBLISHER, št. 9, zv. 4, 2009 , str. 992-1013.

[2] A. K. Dey, "Context-Aware Computing: The CyberDesk Project", v zborniku: *AAAI 1998 Spring Symposium on Intelligent Environments*, Technical Report SS-98-02, 1998, str. 51-54.

[3] A. K. Dey, G. D. Abowd, "Towards a better understanding of context and context-awareness", v zborniku *Technical Report GIT-GVU-99-22*, Georgia Institute of Technology, College of Computing, 1999.

[4] A.K. Dey, G.D. Abowd, A, Wood, "CyberDesk: A Framework for Providing Self-Integrating Context- Aware Services", v zborniku: *Knowledge-Based Systems*, Nov. 1999, str. 3-13.

[5] Alexander N. Pak, Chin-Wan Chung, "A Wikipedia Matching Approach to Contextual Advertising", v zborniku *World Wide Web*, št. 3, zv. 13, 2010, str. 251-274.

[6] A. Ward, A. Jones, A. Hopper, "A New Location Technique for the Active Office", v zborniku *IEEE Personal Communications*, št. 4, zv. 5, 1997, str. 42-47.

[7] B. Schilit, M. Theimer, "Disseminating Active Map Information to Mobile Hosts", v zborniku *IEEE Network*, št. 8, zv. 5, 1994, str. 22-32.

[8] B. Schilit, N. Adams, R. Want, "Context-Aware Computing Applications", v zborniku: *1st International Workshop on Mobile Computing Systems and Applications*, 1994, str. 85-90.

[9] Chen-Chung Liu, Chen-Wei Chung, "Detecting mouse movement with repeated visit patterns for retrieving noticed knowledge components on web pages", v zborniku *Institute of Electronics, Information and Communication Engineers*, št. 10, zv. 90, Okt. 2007, str. 1687-1696.

[10] (2010) devlicio.us, Chrome extension development, dostopno na: http://devlicio.us/blogs/sergio_pereira/archive/2010/06/28/chrome-extension-development.aspx

[11] D. Franklin, J. Flaschbart, "All Gadget and No Representation Makes Jack A Dull Environment" v zborniku *AAAI 1998 Spring Symposium on Intelligent Environments*, Technical Repost SS-98-02, 1998, str. 155-160.

[12] D. Salber, A.K. Dey, G.D. Abowd, "Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm", Georgia Tech GVU Technical Report GIT-GVU-98-01, 1998.

[13] D. Vallet, M. Fernandez, P. Castells, P. Mylonas, Y. Avrithis, "Personalized Information Retrieval in Context", 2006.

[14] (2011) ExtremeTech, Addon's usage, dostopno na: http://www.extremetech.com/internet/87484-85-of-firefox-users-use-add-ons-chrome-users-just-33

[15] (2011) Google, AdSense Help, dostopno na: http://adsense.google.com/support/bin/answer.py?answer=17954

[16] (2011) Google code, Google Chrome Extensions, dostopno na: http://code.google.com/chrome/extensions/getstarted.html

[17] (2011) Google.com, Chrome OS Known Issues, dostopno na: http://www.google.com/support/chromeos/bin/static.py?page=known_issues.cs

[18] (2011) Google Chrome Extensions, dostopno na: http://code.google.com/chrome/extensions/index.html

[19] (2011) Google Chrome Extensions, Overview, dostopno na: http://code.google.com/chrome/extensions/overview.html

[20] (2011) Google Chrome Extensions, xhr, dostopno na: http://code.google.com/chrome/extensions/xhr.html

[21] (2011) Google Search Help Center, Search Features, dostopno na: http://www.google.com/help/features.html

[22] (2009) How to Write a Chrome Extension in Three Easy Steps, dostopno na: http://www.mattcutts.com/blog/write-chrome-extension/

[23] (2011) I Programmer, Internet Explorer loses market share, dostopno na: http://www.i-programmer.info/news/86-browsers/3278-internet-explorer-loses-market-share.html

[24] (2011) It in Canada, Why Google Will Renew Mozilla's (Firefox) Contract, dostopno na: http://www.itincanada.ca/component/kunena/73-the-insight-exchange/2677-why-google-will-renew-mozillas-firefox-contract

[25] J. Pascoe, Adding Generic Contextual Capabilities to Wearable Computers. 2nd International Symposium on Wearable Computers (1998) 92-99.

[26] J. Pascoe, N.S. Ryan, D.R. Morse, "Human-Computer-Giraffe Interaction – HCI in the Field". Workshop on Human Computer Interaction with Mobile Devices, 1998.

[27] L. Han, S. Jyri, J. Ma, K. Yu, "Research on Context-aware Mobile Computing", Proceedings of Advanced Information Networking Applications Workshops, 2008, str 24-30.

[28] Lemur Project, Query Log Toolbar, dostopno na: http://www.lemurproject.org/querylogtoolbar/docs/client/

[29] M. Brown, "Supporting User Mobility", International Federation for Information Processing, 1996.

[30] Merriam-webster dictionary, Context, dostopno na: http://www.merriam-webster.com/dictionary/context?show=0&t=1320160841

[31] M. Henzinger, "Link Analysis in Web Information Retrieval", v zborniku *IEEE Data Engineering Bulletin*, št. 3, zv. 23, Sep. 2000, str. 3-8.

[32] Melike Şah, Wendy Hall, David C De Roure, "SemWeB Semantic Web Browser – Improving Browsing Experience with Semantic and Personalized Information and Hyperlinks", v zborniku *InterFace 2009: 1st National Symposium for Humanities and Technology*, jun. 2009.

[33] (2011) Mozilla addons, Builder and SDK, dostopno na:
https://addons.mozilla.org/en-US/developers/tools/builder

[34] (2011) Mozilla developer, Chrome registration, dostopno na:
https://developer.mozilla.org/en/Chrome_Registration

[35] (2011) Mozilla development, Building an Extension, dostopno na:
https://developer.mozilla.org/en/Building_an_Extension

[36] M.R. Sumalatha, V. Vaidehi, A. Kannan, S. Anandhi "Information Retrieval using Semantic Web Browser Personalized and Categorical Web Search," v zborniku *IEEE-ICSCN 2007*, MIT Campus, Anna University, Chennai, India, feb. 2007, str. 238-243.

[37] N. Ryan, J. Pascoe, D. Morse, V. Gaffney, M. van Leusen, S. Exxon, "Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant", Computer Applications in Archaeology, 1997.

[38] N. Ryan, "Mobile Computing in a Fieldwork Environment: Metadata Elements", Project working document, verzija 0.2, 1997.

[39] (2011) PC Mag, user profile, dostopno na:
http://www.pcmag.com/encyclopedia_term/0,2542,t=user+profile&
i=53562,00.asp#fbid=ibJmEXdlN8l

[40] Ph. Mylonas, D. Vallet, P. Castells, M. Fernsss-a-sssndez, Y. Avrithis "Personalized information retrieval based on context and ontologic knowledge," v zborniku *The Knowledge Engineering Review*, Cambridge University Press, New York, NY, USA, št. 1, zv. 14, mar. 2008 , str. 1-24.

[41] (2010) Piggy Bank, dostopen na:
http://simile.mit.edu/wiki/Piggy_Bank

[42] (2010) Piggy Bank, Save, dostopno na:
http://simile.mit.edu/wiki/Piggy_Bank_Save

[43] (2010) Piggy Bank, Retrieve, dostopno na:
http://simile.mit.edu/wiki/Piggy_Bank_Retrieve

[44] P.J. Brown, "The Stick-e Document: a Framework for Creating Context-Aware Applications" v zborniku *Electronic Publishing '96*, 1996, str. 259-272.

[45] P.J. Brown, "Triggering Information by Context", v zborniku: *Personal Technologies*, št. 2 zv. 1, 1998, str. 1-9.

[46] (2011)Project Hamster, dostopen na: http://projecthamster.wordpress.com/

[47] R. Hull, P. Neaves, J. Bedford-Roberts, Towards Situated Computing, v zborniku: *1st International Symposium on Wearable Computers*, 1997, str. 146-153.

[48] R. Want, A. Hopper, V. Falcao, J. Gibbons, "The Active Badge Location System", v zborniku *ACM Transactions on Information Systems*, št.1, zv. 10, 1992, str. 91-102.

[49] Schilit, B., Theimer, M. Disseminating, "Active Map Information to Mobile Hosts", v zborniku *IEEE Network*, št. 5, zv. 8, 1994, str.22-32.

[50] S. Greenberg, "Context as a dynamic construct", v zborniku *Human-Computer Interaction*, Publisher L. Erlbaum Associates Inc, št. 16, zv. 2, 2001 , str. 257-268.

[51] S. Fickas, G. Korteum, Z. Segall, "Software Organization for Dynamic and Adaptable Wearable Systems", v zborniku: *1st International Symposium on Wearable Computers*, 1997, str. 56-63.

[52] Vandana Dhingra, Dr. Komal Kumar Bhatia, "Towards Intelligent Information Retrieval on Web," v zborniku *International Journal on Computer Science and Engineering*, št. 4, zv.3, apr. 2011, str. 1721-1726.

[53] (2011) The Lemur project, Query Log Toolbar, dostopen na: http://www.lemurproject.org/

[54] (2011) The Lemur project, Query Log Toolbar, vtičnik za Firefox, dostopen na: http://lemur.sourceforge.net/querylogtoolbar/querylogtoolbar.xpi

[55] (2009) Techie Buzz, Google Chrome Extensions vs. Firefox Addons, dostopno na: http://techie-buzz.com/featured/google-chrome-extensions-vs-firefox-addons.html

[56] (2011) Techtarget, user profile, dostopno na: http://searchsecurity.techtarget.com/definition/user-profile

[57] T. Rodden, K. Cheverst, K. Davies, A. Dix, "Exploiting Context in HCI Design for Mobile Systems" Workshop on Human Computer Interaction with Mobile Devices, 1998.

[58] (2010) Web Browser Grand Prix: The Top Five, Tested And Ranked, dostopno na: http://www.tomshardware.com/reviews/firefox-chrome-opera,2558-3.html

[59] (2011) Web Browser Showdown: Memory Management Tested, dostopno na: http://www.tested.com/news/web-browser-showdown-memory-management-tested/2420/

[60] (2011) Wikipedia, Context awareness, dostopno na: http://en.wikipedia.org/wiki/Context_awareness

[61] Wikipedia, Epistemology, dostopno na: http://en.wikipedia.org/wiki/Epistemology

[62] 2011) Wikipedia, Gecko, dostopno na: http://en.wikipedia.org/wiki/Gecko_%28layout_engine%29

[63] (2011) Wikipedija, Google Chrome, dostopno na: http://en.wikipedia.org/wiki/Google_Chrome

[64] Wikipedia, Great Architcture for Text Engineering, dostopno na: http://en.wikipedia.org/wiki/General_Architecture_for_Text_Engineering

[65] Wikipedia, Legalism, dostopno na: http://en.wikipedia.org/wiki/Legalism_%28Chinese_philosophy%29n

[66] (2011) Wikipedia, Lemur Project, dostopno na: http://en.wikipedia.org/wiki/Lemur_Project

[67] (2008) Wikipedija, Mashup, dostopno na: http://en.wikipedia.org/wiki/Mashup_%28web_application_hybrid%29

[68] (2011) Wikipedia, User profile, dostopno na: http://en.wikipedia.org/wiki/User_profile

[69] Wikipedia, Resource Description Framework, dostopno na: http://en.wikipedia.org/wiki/Resource_Description_Framework