Peräkkäishahmojen tunnistaminen webin käytön louhinnassa

Sami Yläkäs

Helsinki 7.4.2008

Tiedon louhinnan seminaari, kevät 2008

Seminaaritutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Sami Yläkäs

Työn nimi — Arbetets titel — Title

Peräkkäishahmojen tunnistaminen webin käytön louhinnassa

Oppiaine — Läroämne — Subject		
Tietojenkäsittelytiede		
Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaaritutkielma	7.4.2008	14 sivua

Tiivistelmä — Referat — Abstract

Tämän tutkielman tarkoitus on antaa lukijalle yleiskuva webin käytön louhinnasta lyhyesti, esitellä kaksi yleisintä algoritmiluokkaa peräkkäishahmojen tunnistamiseksi, ja vertailla tämän hetken tehokkaimpia algoritmeja keskenään. Peräkkäishahmojen tunnistamiseen soveltuvilla algoritmeilla louhitaan usein toistuvia sekvenssejä web-palvelimen lokitiedostosta. Louhitut peräkkäishahmot ovat hyödyllisiä, kun niitä lopuksi analysoidaan, mikä auttaa esimerkiksi virheellisten navigointipolkujen löytämisessä ja sitä kautta sivurakenteen parantamisessa.

Erityisesti WAP-puu-algoritmia (Web Access Pattern) esitellään yksityiskohtaisesti. WAP-puulla on hyvin tiivis ja järjestetty tietorakenne – se tallentaa lokitiedon prefix-muotoon samalla tavalla kuin FP-puu (Frequent Pattern), jossa tietoalkioiden järjestykseen ei oteta kantaa. Peräkkäishahmojen perusominaisuus on järjestykseen sekä aikaan sidonnaisuus, ja tapahtumat voivat esiintyä uudelleen samassa sekvenssissä. Perinteinen WAP-puu käy läpi sekvenssitietokannan vain kahdesti ja välttää kasvavien kandidaattijoukkojen luomisen, mikä on ongelma Apriori-tyyppisissä algoritmeissa, kun tietolähde on suuri. Webin lokitiedostot luonnollisesti kasvavat usein pitkiksi, joten algoritmin on hyvä olla tehokas suurella tietomäärälläkin. WAP-puusta on kehitetty ainakin kaksi uudempaa ja tehokkaampaa versiota, jotka toimivat ilman välipuiden rakentamisia. Puurakenteiden lisäksi mainitaan lyhyesti myös Markovin ketjut.

Avainsanat — Nyckelord — Keywords
Webin käytön louhinta, peräkkäishahmot, WAP-puu, Markovin ketjut

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Webistä voidaan louhia niin sivujen sisältöä, niiden ja verkon rakennetta kuin webin käyttöäkin. Tässä työssä käsitellään käyttäjän navigointia linkkien välillä, transaktioita ja käyttäytymistä webissä.

Termin webin käytön louhinta esitteli ensimmäiseksi Cooley et al. [CMS97]. He tutkivat käyttäjien ominaisuuksien oppimista ja ennustamista Internetissä. Webin käytön louhinta voidaan yleisesti jakaa kolmeen päävaiheeseen: tiedon esikäsittely, hahmojen etsintä ja löydettyjen hahmojen analysointi. Lokeista erotettavia käyttöhahmoja voidaan soveltaa esimerkiksi webin personalisointiin ja sivurakenteen parantamiseen. Webin kehityksen sanotaan olevan jo toisessa sukupolvessa [SLBR07], mikä tarkoittaa yhteisöllisempää sisällön tuottamista ja jakelua, sekä niiden taustalla olevia uusia teknologioita. Varsinkin Ajax tekee webin käytön louhinnasta erittäin haastavan tehtävän, koska totuttujen hyperlinkkien sijasta käytetään yhä enemmän XMLHttpRequest-pyyntöjä ja sivua renderöidään osittain.

Yleensä palvelimen omat lokitiedostot eivät tarjoa suhteellista tietoa eri ajanjaksojen välillä, vaan ovat vain lista erityyppisistä muuttujista: pyydetty sivu parametreineen, sivupyynnön ajankohta, ja hakuun käytetty aika. Selain voi jättää ja palvelin tallentaa hyvin yksityiskohtaisiakin tietoja ympäristömuuttujiin, joilla voidaan identifioida käyttäjä IP-osoitteen sekä evästeiden avulla. Tässä raportissa tutkitaan pääasiassa staattisten lokitietojen louhintaa peräkkäishahmojen tunnistamiseksi, ja lopussa viitataan myös dynaamisen louhinnan mahdollisuuksiin. On olemassa kaksi tärkeää algoritmiluokkaa: toinen pohjautuu assosiaatiosääntöihin, ja toinen puurakenteisiin sekä Markovin ketjuihin.

2 Tietolähteet ja webin käytön louhinnan prosessi

Webin käytön louhinnan sovellukset perustuvat kolmeen eri tietolähteeseen, joita ovat web-palvelimet, proxy-palvelimet, ja asiakkaan selainpohjaiset lähteet. Web-palvelimen lokitiedostot ovat nykyisissä malleissa yleisin sekä suurin tietolähde. Lokitiedostoformaatti riippuu palvelimen asetuksista. Muotoja on olemassa useita, eräs standardimuoto on NCSA Common, joka sisältää perustiedot sivupyynnöstä (Taulukko 1).

Kun lokitietoja hyödynnetään, pääasiallisena ongelmana on käyttäjien sessioiden tunnistaminen, eli kuinka voidaan ryhmitellä kaikkien käyttäjien sivupyynnöt siten,

Kenttä	**Kentän arvo**
Asiakkaan IP-osoite	125.125.125.125
Käyttäjän tunniste	-
Käyttäjän nimi	ylakas
Päivämäärä ja aika	[28/Feb/2008:20:15:05 +0300]
HTTP-pyyntö	”GET /index.html HTTP/1.0”
Vastauksen statuskoodi	200
HTTP-pyynnön tavumäärä	1043

Taulukko 1: NCSA Common -tiedostomuoto.

että niistä voidaan selkeästi tunnistaa käyttäjän hiirellä klikatut navigointipolut. Käyttäjällä tarkoitetaan yleensä oikeaa hiirellä navigoivaa henkilöä, mutta lokitiedoissa voi esiintyä useasti myös selainta jäljittelevien ohjelmien jälkiä. Hakukoneet esim. Google, jättävät web-palvelimen lokeihin tiedon sivulla vierailusta. Lisäksi on olemassa monia muita robotteja, jotka keräävät tietoa eri käyttötarkoituksiin. Itseasiassa niitä voidaan pitää esimerkkinä webin rakenteen louhinnan työkaluista, kun halutaan erottaa ihmiskäyttäjien navigointihahmot robottien hahmoista.

Session eli käyttäjän istunnon tunnistaminen on usein vaikeaa, koska palvelimelta saatava tieto on hyvin vaihtelevaa. Epäolennaisen tiedon poistaminen voidaan toteuttaa tutkimalla URL:n loppuliitteitä. Paikallinen välimuisti ja proxy-palvelimet voivat rikkoa kokonaiskuvaa käyttäjän navigoinnista. Sivu, joka on kirjoitettu vain kerran lokiin, voi itseasiassa olla viitattuna monta kertaa useammalla käyttäjällä. Ongelman ratkaisemiseksi voidaan ottaa avuksi seuraavat menetelmät:

- Käytetään evästeitä (cookies), jotka tallennetaan tilapäisesti yksittäisen istunnon ajaksi välimuistiin tai pidemmäksi aikaa käyttäjän tietokoneelle, mikä helpottaa istuntojen seurantaa.
- Ei käytetä välimuistia, jolloin kaikki sivupyynnöt menevät palvelimelle asti.
- Luodaan käyttäjätunnuksella ja salasanalla yksilöivä sessiotunnite.

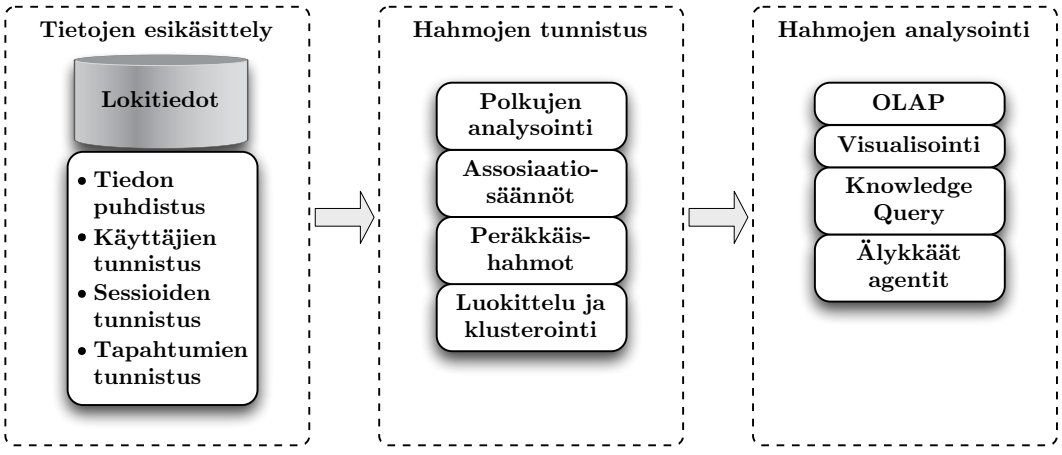
Toinen ongelma proxy-palvelimissa on käyttäjän tunnistaminen. Koneen nimen käyttäminen identifioivana tietona voi johtaa useiden käyttäjien ryhmän pitämiseen yhtenä käyttäjänä. Jos evästeitä ei ole saatavilla, voidaan joissain tapauksissa käyttää

myös heuristisia menetelmiä. Vaikka evästeiden avulla saadaankin käyttäjä konekohtaisesti tunnistettua, lokitietoihin ei jää selaimen Back-painikkeen klikkauksia, koska nämä pyynnöt kohdistuvat vain selaimen välimuistiin.

Asiakkaan puolella käyttötietoa voidaan seurata JavaScriptillä sekä muilla asiakaspuolen skriptikielillä. Myös modifioidut selaimet ovat yksi tapa lähettää käyttäjän navigointitietoa palvelimelle. Asiakaspuolen ratkaisut välttävät välimuistista aiheutuvan käyttäjän tunnistusongelman.

Harvoin käytetty, mutta mahdollinen tapa seurata käyttäjien navigointia on TCP/IP-pakettien tarkkaileminen verkkoliikenteessä. Etuna on reaaliaikaisuus, mutta sovelluksissa, joissa liikenne on SSL-kryptattua, tämä menetelmä on käyttökelvoton. Ehkäpä parempi lähestymistapa webin käytön seuraamiselle on suoraan sovellustasolla, mutta se tarkoittaisi sitä, että jokaista sovellusta kohden tulisi olla oma jäljitysjärjestelmä.

Lokin analysointia pidetään yksinkertaisimpana menetelmänä webin käytön louhinnassa. Usein se on myös käyttökelpoisin tapa, koska analysoija on tyypillisesti palvelun tarjoaja, ja lokitieto on aina helposti saatavilla. Webin käytön louhinnan tarkoituksena on soveltaa tilastotieteen ja tiedon louhinnan menetelmiä esikäsiteltyyn webin lokidataan, josta voidaan tunnistaa hyödyllisiä hahmoja. Tilastoanalyysi on yleisin metodi analysointiin. Tiedon louhinnan metodeja ovat assosiaatiosääntöjen ja peräkkäishahmojen tunnistaminen. Klusterointi ja luokittelu ovat myös tärkeitä menetelmiä, mutta ne jätetään tämän työn ulkopuolelle. Kuvassa 1 on esitetty pääpiirteittäin webin käytön louhinnan osa-alueet ja prosessin kulku. Tämä työ keskittyy kuvan keskimmäiseen osaan ja peräkkäishahmojen louhintaan.



Kuva 1: Webin käytön louhinnan prosessi.

3 Peräkkäishahmojen tunnistus

Webin käytön louhinnassa peräkkäishahmoilla(sequential patterns) voidaan löytää kiinnostavia navigointipolkuja sivujen välillä. Peräkkäishahmot saattavat ilmetä syntaktisesti samanlaisina assosiaatiosääntöihin verrattuna. Algoritmeja, joita käytetään assosiaatiosäännöissä, voidaan myös käyttää peräkkäishahmojen louhinnassa. Peräkkäishahmot kuitenkin sisältävät tietoa tapahtumien järjestyksestä.

On olemassa kaksi tärkeää algoritmiluokkaa peräkkäishahmojen tunnistamiseksi: toinen pohjautuu assosiaatiosääntöihin ja toinen puurakenteisiin sekä Markovin ketjuihin navigointihahmojen esittämiseksi [FL05]. Markovin malli on tyypillinen stokastinen menetelmä.

3.1 Määritelmiä

Lokitietoja voidaan ajatella kokoelmana ajan mukaan järjestettyjä tapahtumasarjoja, joita syntyy käyttäjän vieraillessa WWW-sivustolla. Esikäsittelyllä saadaan raaka-aineistosta tiiviimmässä muodossa oleva tietorakenne. Lähteessä [AS95] on kuvattu kattavasti peräkkäishahmojen louhintamekanismi. Sekvenssitietokannassa, jossa jokainen sekvenssi on lista tapahtumia aikajärjestyksessä, ja jokainen transaktio sisältää joukon tietoalkioita, on löydettävä peräkkäishahmot käyttäjän määrittelemällä minimituella, jossa tuki on hahmon sisältävien tietosekvenssien lukumäärä.

Määritelmä 1. *Olkoon E joukko yksittäisiä pyyntöjä, jotka esittävät käyttäjien hakemia web-tietolähteitä. $S = e_1 e_2 \dots e_n$ ($e_i \in E$, $1 \leq i \leq n$) on tapahtumien pyyntösekvenssi, ja $|S| = n$ on S:n pituus.*

Määritelmä 2. *Sekvenssissä $S = e_1 e_2 \dots e_k e_{k+1} \dots e_n$, $S_{prefix} = e_1 e_2 \dots e_k$ on S:n alkuosa ja $S_{suffix} = e_{k+1} e_{k+2} \dots e_n$ on S:n loppuosa.*

Web-pyyntösekvenssi voidaan merkitä $S = S_{prefix} + S_{suffix}$. Kaikki web-pyyntösarjat tietokannassa voivat kuulua yhdelle tai useammalle käyttäjälle. Oletetaan, että on joukko pyyntösekvenssejä tapahtumajoukolla $E = \{a, b, c, d, e, f\}$. Silloin esimerkiksi $S = afbacfc$ voidaan muotoilla myös $S = a + fbacfc = af + bacfc = \dots = afbacf + c$.

Määritelmä 3. *Olkoon S_1 ja S_2 sarjan S loppuosia, ja olkoon S_1 lisäksi S_2:n loppuosa. Silloin S_1 on S_2:n aliloppusekvenssi (sub-suffix sequence).*

Olkoon sekvenssitietokanta $WAS_{DB} = \{S_1, S_2, \dots, S_m\}$, jossa S_i ($1 \leq i \leq m$) on pyyntösekvenssi, ja $|WAS_{DB}| = m$ tietokannan koko. Tuki S:lle tietokannassa

WAS_{DB} määritellään yhtälöllä (1) [ZHF06].

$$sup\,(S) = \frac{|\{S_i \mid S \subseteq S_i, S_i \in WAS_{DB}\}|}{|WAS_{DB}|} \quad (1)$$

Pyyntösekvenssiä S sanotaan *peräkkäispyyntöhahmoksi* (sequential access pattern) [ZHF06], jos $sup\,(S) \geq MinSup$, missä $MinSup$ on minimituki.

Hyvin tuoreessa lähteessä [MPTM08] määritelmät puolestaan nojautuvat suoraan lokitiedostomuotoon (esim. luvussa 2 mainittu NCSA). Lokitiedosto prosessoidaan kahdessa vaihessa. Ensin tiedosto järjestetään osoitteen ja transaktion mukaan, jonka jälkeen tiedostosta karsitaan pois epäkiinnostava tieto. Järjestyksen aikana URL-osoitteet ja asiakkaat käsitellään kokonaislukuina, lisäksi päivämäärät ja kellonajat muunnetaan suhteellisiksi ensimmäiseen aikaan lokitiedostossa.

Määritelmä 4. *Olkoon L joukko merkintöjä palvelimen lokitiedostossa. Merkintä g, $g \in L$, on alkiopari $g = \langle ip_g, ([l_1^g \cdot URL,\ l_1^g \cdot time] \ldots [l_m^g \cdot URL,\ l_m^g \cdot time\,])\rangle$ siten, että välillä $1 \leq k \leq m$, $l_k^g \cdot URL$ on käyttäjän g pyytämä tietoalkio ajankohtana l_k^g ja kaikille $1 \leq j < k$, $l_k^g \cdot time > l_j^g \cdot time$.*

Jotta lokitiedostosta voitaisiin löytää usein toistuvia hahmoja, jokaiselle g:lle on muunnettava ip_g asiakasnumeroksi, ja jokaiselle tietueelle k, $l_k^g \cdot time$ muutetaan aikanumeroksi ja $l_k^g \cdot URL$ muutetaan tietoalkionumeroksi. Taulukossa 2 on esimerkki tiedostosta esikäsittelyn jälkeen. Jokaiselle asiakkaalle on olemassa vastaava aikasarja, jossa on mukana asiakkaan pyytämä URL. Esimerkiksi, asiakas 2 pyytää URL:n U_6 hetkellä $d4$. Asiakkaan 1 lokimerkintä tässä taulukossa on: $E_1 = \langle c_1,\ ([U_1, d_1]\ [U_3, d_2]\ [U_4, d_3]\ [U_2, d_4]\ [U_3, d_5])\rangle$.

Peräkkäishahmojen tunnistamisessa tavoitteena on löytää tiedostosta hahmoja, joita voidaan pitää usein toistuvina. Taukosta 2 voidaan havaita, että minimituella 100% saadaan tulos $\langle (U_1)\,(U_3)\,(U_2)\,(U_3)\rangle$.

Asiakas	**d1**	**d2**	**d3**	**d4**	**d5**
1	U_1	U_3	U_4	U_2	U_3
2	U_1	U_3	U_2	U_6	U_3
3	U_1	U_7	U_3	U_2	U_3

Taulukko 2: Tiedosto esikäsittelyn jälkeen [MPTM08].

3.2 Assosiaatiosäännöt

Peräkkäishahmot ovat assosiaatiosääntöjen yleistys, joka ilmaisee hahmojen samankaltaisuutta tietyllä ajanjaksolla. Webissä sellainen hahmo voi olla sivu tai sivujoukko, joilla on vierailtu heti toisen sivujoukon jälkeen. Tätä tapaa käyttäen voidaan tunnistaa käyttäjäkohtaisia trendejä ja ennustaa seuraavia sivujoukkoja.

Assosiaatiosääntöjen avulla voidaan löytää toistuvia hahmoja, assosiaatiota ja korrelaatioita tietoalkoiden välillä. Niillä voidaan osoittaa mahdollisia riippuvuuksia sivujen välillä, joilla ollaan käyty useasti, vaikka ne eivät olisikaan suoraan toisiinsa kytköksissä. Assosiaatiosäännöt myös ilmaisevat assosiaatioita tietyn tyyppisten käyttäjäryhmien välillä. Tyypillisellä perkkäishahmolla on seuraavan muoto: 70% käyttäjistä, jotka ensin kävivät sivulla *A.html* ja sen jälkeen sivulla *B.html*, ovat myös vierailleet sivulla *C.html* saman session sisällä [FL05]. Edellä mainitussa esimerkissä sivut A, B ja C esiintyvät peräkkäin, toinen toisensa jälkeen, kun taas assosiaatiomallissa tapahtumien järjestykseen ei oteta mitään kantaa.

Määritelmä 5. *Olkoon* $I = \{i_1, i_2, \ldots, i_m\}$ *joukko alkioita ja* $D = \{t_1, t_2, \ldots, t_n\}$ *joukko transaktioita; jokaisella transaktiolla on yksilöivä tunniste TID ja tietoalkiojoukko* I*.* I *on* k*:n kokoinen alkiojoukko. Transaktio* T *sisältää* X*:n, joukon joitain* I*:n alkioita, jos* $X \subseteq T$*.*

Assosiaatiosääntö merkitään implikaationa $I_1 \Rightarrow I_2$, missä I_1 ja I_2 ovat I:n osajoukko ja niillä ei ole yhteisiä alkioita eli $I_1 \cap I_2 = \emptyset$. Kiinnostavat assosiaatiosäännöt erotellaan koko joukosta tuella s tai luottamuksella c. Tuki kertoo kuinka usein I_1 ja I_2 esiintyvät yhdessä. Luottamus puolestaan määrittää säännön todennäköisyyden. Assosiaatiosääntöjen löytämiseksi on olemassa monia algoritmeja, joista tunnetuin lienee Apriori. Peräkkäishahmojen tunnistamiseen soveltuvista algoritmeista mainittakoon AprioriAll ja GSP.

3.3 Puurakenteet

Assosiaatiosääntöihin perustuvilla algoritmeilla on yhteinen ongelma. Ne vaativat työlästä tietolähteen skannausta useaan kertaan, kun tarkastellaan pitkiä peräkkäishahmoja. Peräkkäishahmojen tunnistamisen tehostamiseksi, Pei et al. [PHMAZ00] ovat esittäneet WAP-puun (Web Access Pattern), joka perustuu FP-puuhun [HPY00]. Algoritmissa ei generoida suurta määrää kandidaatteja kuten assosiaatiosäännöissä. Luotu WAP-puu on usein paljon pienempi kuin alkuperäinen sekvenssitietokanta. Kokeelliset tulokset ovat osoittaneet, että algoritmi on yleisesti nopeampi kuin pe-

rinteiset peräkkäishahmojen tunnistusmenetelmät.

Oletetaan, että minimituen raja-arvo on 75%. WAP-puun rakentamiseksi on ensin skannattava tietokanta kerran läpi, jotta saadaan usein toistuvat tapahtumat. Puun rakennusvaiheessa vähiten toistuvat osat jokaisessa jonossa poistetaan, ja vain usein toistuvia alisekvenssejä käytetään syötteenä. Esimerkkitaulukossa 3 kaikille tapahtumille a, b, c, d, e, f voidaan määrittää tuet $4, 4, 4, 1, 2, 3$ vastaavassa järjestyksessä. Minimituella 3 vain a, b ja c ovat usein toistuvia tapahtumia. Siispä kaikki toistumattomat tapahtumat (d, e, f) poistetaan jokaisesta transaktiosekvenssistä, jolloin saadaan usein toistuvista alkioista koostuva alisekvenssi taulukon 3 kolmannessa sarakkeessa.

TID	**Pyyntösekvenssi**	**Kattava alisekvenssi**
100	abdac	abac
200	eaebcac	abcac
300	babfaec	babac
400	afbacfc	abacc

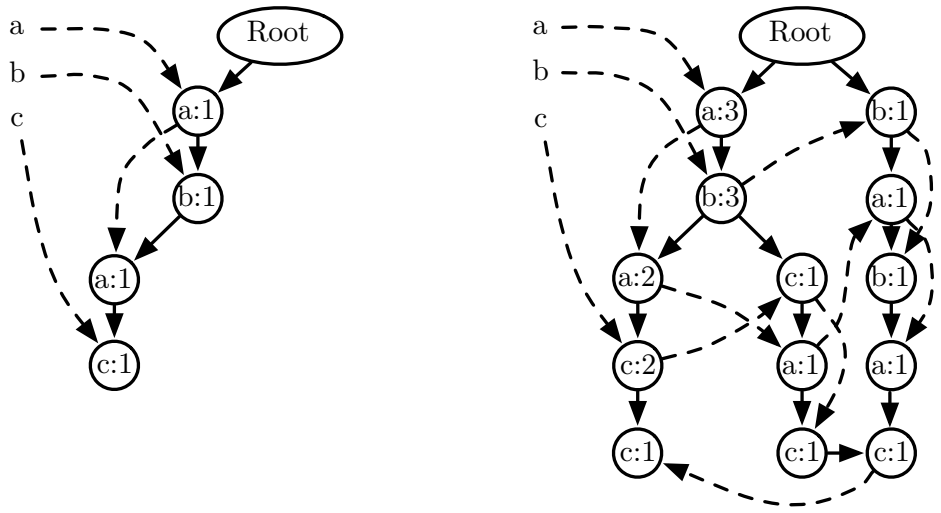
Taulukko 3: Pyyntösekvenssit WAP-puulle [EL05].

Jokaisen transaktion kattavalle sekvenssille WAP-puu-algoritmi tallentaa ensimmäiseksi usein toistuvat alkiot otsikkosolmuiksi (a, b, c), joita käytetään kaikkien tämän tyyppisten solmujen linkittämiseen WAP-puussa lisäämisjärjestyksessä.

Kun WAP-puuta rakennetaan, asetetaan ensimmäiseksi virtuaalinen juuri. Tämän jälkeen jokaista kattavaa sekvenssiä transaktiossa käytetään puun polkujen rakentamiseen juuresta lehtisolmuihin. Jokainen tapahtuma sekvenssissä lisätään uutena solmuna juuresta alkaen, jos kyseisellä tapahtumalla ei ole vielä olemassa solmua. Lisäksi lisätylle solmulle liitetään kaari katkoviivalla viimeisimmästä saman tyyppisestä lisätystä solmusta. Jos kyseinen tapahtuma on vasta ensimmäinen, lisätään uudelle solmulle kaari saman tyyppisestä otsikkosolmusta. Esimerkkinä kuva 2(a), jossa lisätään ensimmäinen kattava sekvenssi *abac*, joka kuuluu transaktiolle ID 100 esimerkkitietokannassa. Koska solmua a ei ole aivan aluksi olemassa, luodaan juuren vasen lapsisolmu a laskurin arvolla 1. Sen jälkeen liitetään kaari katkoviivalla lisättyyn solmuun a otsikkosolmusta. Seuraava tapahtuma b lisätään solmun a lapsisolmuksi myös laskurin arvolla 1 ja liitetään otsikkosolmuun b. Kolmas tapahtuma a lisätään solmun b lapseksi laskurilla 1, ja tälle uudelle a:lle asetetaan kaari viimeisimmästä lisätystä a:sta. Neljäs ja viimeinen tapahtuma tässä sekvenssissä on c, ja

(a) WAP-puu
(tapahtumien *abac* lisäämisen jälkeen)

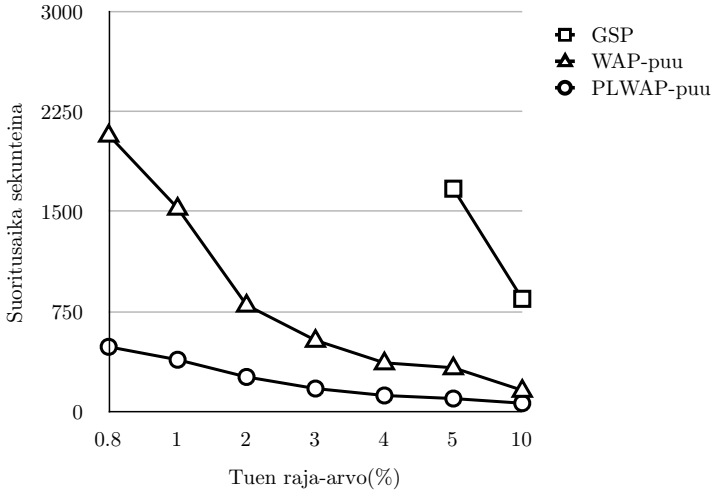
(b) Täydellinen WAP-puu



Kuva 2: WAP-puun konstruointi [EL05].

se lisätään viimeksi lisätyn a:n lapsisolmuksi tässä polussa. Lehtisolmu c:lle lisätään myös kaari otsikkosolmusta c.

Samalla tavalla lisätään seuraava sekvenssi *abcac* ID:llä 200, alkaen virtuaalisesta juuresta. Koska juurella on jo lapsi a, solmun a tapahtumalaskuria kasvatetaan yhdellä, mistä saadaan $a : 2$. Samoin seuraavalle tapahtumalle b löytyy jo solmu, joten kasvatetaan senkin laskurin arvoksi 2. Tapahtuma c ei vastaa seuraavaa olemassa olevaa solmua a, joten uusi solmu $c : 1$ luodaan ja se lisätään solmun b toiseksi lapseksi. Kolmas sekvenssi *babac* ja neljäs sekvenssi *abacc* lisätään samalla menetelmällä, jolloin saadaan lopputulokseksi kuvan 2(b) mukainen täydellinen WAP-puu, mikä sisältää kaikki sekvenssit. Valmista WAP-puuta louhitaan usein toistuvien hahmojen löytämiseksi alkaen vähiten kattavasta tapahtumasta otsikkolistassa. Esimerkissä kyseinen tapahtuma on c. Algoritmi on kokonaisuudessaan kuvattu lähteessä [EL05]. Huonona puolena WAP-puu-algoritmeissa on se, että ehdollisten hakustrategioiden käyttäminen vaatii useita välipuiden uudelleenrakentamisia, mikä ei ole tehokasta. CSB-mine-algoritmi [ZHF06], välttää WAP-puiden uudelleenrakentamisen ja voi parantaa suoritusta erityisesti pienillä tuen raja-arvoilla, kun tietokanta on suuri. WAP-puuta ovat parannelleet myös Ezeife et al. [EL05]. Heidän algoritminsa toimii myös ilman välipuiden rakentamisia ja tulosvertailuissa on mukana aikaisemmin mainittu assosiaatiosääntöjä käyttävä GSP-algoritmi (Kuva 3).



Kuva 3: Suoritusajat erilaisilla minimituilla [EL05].

3.4 Markovin ketjut

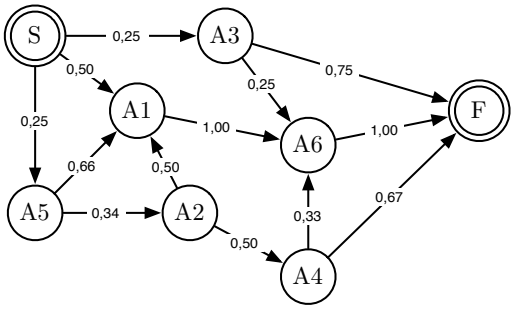
Webin navigointipolkuja voidaan esittää myös ns. HPG-mallilla [JPT02], josta on esimerkki kuvassa 4. Siinä esitetään siirtymiä WWW-sivujen välillä, missä on samoja piirteitä kuin Markovin ketjuissa. Samat tutkijat ovat myöhemmin arvioineet Markov-mallien olettamusta [JPT03], jonka mukaan seuraava sivupyyntö on vain riippuvainen edellisistä sivypyynnöistä (kuva 5). He osoittavat, että tämä ei ole aina totta, ja Markovin malleihin perustuvat rakenteet webin käytön louhinnassa soveltuvat parhaiten vähemmän tarkkuutta vaativiin tehtäviin, kuten esimerkiksi personalisointiin ja kohdennetuihin mainoksiin.

Diskreetti Markovin ketjumalli [Sar00] voidaan määritellä alkiokolmikkona $\langle S, \boldsymbol{A}, \lambda \rangle$. S vastaa tila-avaruutta, $\boldsymbol{A}$ on matriisi, joka esittää tilasiirtymien todennäköisyyttä, ja λ on alkutilojen todennäköisyysjakauma avaruudessa S. Markovin mallin perusominaisuus on tilan riippuvuus vain edellisestä tilasta. Jos vektori $\vec{\boldsymbol{s}}(t)$ esittää todennäköisyyttä kaikille tiloille ajankohtana t, niin:

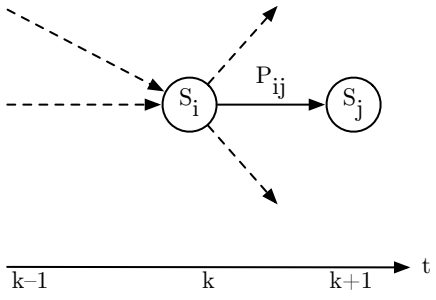
$$\vec{\boldsymbol{s}}(t) = \vec{\boldsymbol{s}}(t-1)\boldsymbol{A}. \tag{2}$$

Jos on olemassa n tilaa, siirtymätodennäköisyysmatriisi $\boldsymbol{A}$ on kokoa $n \times n$. Markovin ketjuja voidaan soveltaa peräkkäisten sivuhakujen mallintamiseen. Tällöin Markovin mallin tilakäsite voi olla URL, HTTP-pyyntö tai toiminta (esimerkiksi tietokannan

päivitys tai sähköpostin lähetys). Matriisin $\boldsymbol{A}$ siirtymätodennäköisyyksiä voidaan arvioida useilla menetelmillä [Sar00].



Kuva 4: Esimerkki HPG [JPT03].



Kuva 5: Markovin olettamus [JPT03].

4 Hahmojen käyttökohteet

Hahmojen analysointi on viimeinen vaihe koko webin käytön louhinnan prosessissa. Saatavilla on louhittu tieto, josta halutaan poistaa epäolennaiset hahmot, joita löydettiin hahmojen tunnistusvaiheessa. Tarkka analysointi suoritetaan yleensä sovelluksella, jolle webin käytön louhinta on tehty. Yleisimmin tunnettu hahmon analysointi sisältää ”Knowledge Query”-mekanismin, kuten SQL:n. Toinen tapa on ladata käyttötietoa datakuutioon OLAP-operaatioiden (On-Line Analytical Processing) suorittamiseksi [LW04]. Visualisointitekniikat, kuten graafiset hahmot ja värien asettelu, voivat yleensä tuoda esiin kokonaisvaltaisia hahmoja ja trendejä datassa [CHM+03]. Sisältö- ja rakennetietoa voidaan käyttää suodattamaan hahmoja, jotka esittävät jotain käyttötapaa, sisällön tyyppiä, tai sivuja, jotka vastaavat tiettyä hyperlinkkirakennetta.

Muita sovelluskohteita ovat web-robottien tunnistaminen [TK02], käyttäjän virheellisten navigointipolkujen löytäminen ja sitä kautta sivurakenteen parantaminen [TKK05], käyttäytymishahmojen tunnistaminen ja reittien yksinkertaistaminen [DvDD07] sekä käyttäjän personalisoinnin sovellukset [EV07, MDLN01].

Tietoturva ja käyttäjän yksityisyys webissä on myös huomioon otettava asia, kun profiloidaan käyttäjiä, koska käyttäjä ei välttämättä ole lainkaan tietoinen, että esim. ostokäyttäytymisestä voidaan kerätä tietoa markkinointitarkoituksiin, tai pahimmassa tapauksessa tietoa voidaan käyttää laittomasti. On olemassa EU-direktiivi, joka sisältää säännöt tietojen tallentamisesta (esim. evästeet) käyttäjän tietokoneelle: Tiedon tallentaminen on sallittua vain, jos käyttäjää informoidaan kuinka

tallennettua tietoa käytetään ja käyttäjälle annetaan mahdollisuus hylätä tietojen tallennus. Lisäksi mainitaan, että säännöt eivät koske järjestelmää, missä on teknisten syiden vuoksi on pakko käyttää tiedon tallentamista. Tämähän on yleinen tapa esim. pankkipalveluissa sekä muissa järjestelmissä, joissa henkilö tunnistetaan käyttäjätunnus–salasana-yhdistelmällä. Useimmissa uusimmissa selaimissa voi estää kolmannen osapuolen mainoksissa olevat evästeet, eli hyväksyä vain evästeet, jotka tulevat samalta palvelimelta kuin haettu sivu.

5 Yhteenveto

Tässä raportissa käsiteltiin webin käytön louhintaa peräkkäishahmojen avulla, tarkasteltiin kahta yleisintä algoritmiluokkaa hahmojen tunnistamiseksi, vertailtiin niiden hyviä sekä huonoja puolia ja todettiin, että puumallit sekä Markovin ketjuihin perustuvat algoritmit suoriutuvat paremmin kuin perinteiset assosiaatiosääntöjä käyttävät menetelmät. Tietolähteen tyyppi ja datan koko vaikuttavat siihen, mikä algoritmi on missäkin tilanteessa optimaalisin. Raportissa mainitut menetelmät ovat vain murto-osa siitä määrästä tiedon louhinnan algoritmeista, joita voidaan hyödyntää peräkkäishahmojen tunnistamiseen. Eräs mielenkiintoinen lähestymistapa on jäljitellä muurahaisten käyttäytymistä polkujen valinnassa, missä tutkimusten mukaan on hiukan parempi ennustustarkkuus kuin Markovin mallissa lyhyillä session pituuksilla [LLY07].

Yhteistä kaikissa menetelmissä on se, että ne käyttävät web-lokitiedostoja tietolähteenä. Tutkimustuloksia selainpohjaisesta webin käytön louhinnasta ei juurikaan löydy. Lähteessä [RZP07] on tutkittu dynaamista louhintaa, mutta siinä käytetty FTS-louhinta (Frequent Traversal Sequence) on terminä harvinainen. Webin kehitys on menossa suuntaan, missä klassinen sivumalli, jossa siirrytään linkkien kautta sivulta toiselle, voi tulevaisuudessa olla vain jäänne vanhasta teknologiasta. Saman sivuston sisällä pelkkien lokitietojen tutkiminen voi tulla ongelmaksi, jos sivulla on käytetty Ajax-ohjelmointia, mikä on nopeasti yleistymässä yritysmaailman web-sovelluksissa. Silloin asiakkaan sivupyynnöt eivät välttämättä tarkoita kokonaan uuden sivun lataamista palvelimelta, mikä jäisi lokitietoihin, vaan useat sivupyynnöt säilyttävät URL-osoitteen ja ovat itseasiassa vain viestejä palvelimelle käyttäjän tapahtumista. Jo nykytekniikalla on mahdollista ohjelmoida sovellus, jossa URL-osoite ei vaihdu koskaan. Tapahtumaa ei voida päätellä lokitietoa lukemalla, koska ensinnäkin vain sovellus itse tietää mitä se antaa vastauksena tietylle käyttäjän tapahtumalle, ja

toiseksi, nämä tapahtumapyynnöt ja parametrien arvot voivat olla salattu sovelluksen omalla avaimella ilman HTTPS-protokollaakin. Nähtäväksi jää mihin suuntaan web-selaimet kehittyvät, koska nykymallilla ne eivät ole HTML-standardien takia miellyttävin ohjelmointialusta. Ehkäpä XAML (Extensible Application Markup Language) tai XUL (XML User Interface Language) -sivut ovat tulevaisuudessa Googlen hakutulosten kärjessä.

Lähteet

AS95 Agrawal, R. ja Srikant, R., Mining sequential patterns. *Proc. of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, Yu, P. S. ja Chen, A. L. P., toimittajat. IEEE Computer Society, 1995, sivut 3–14.

CHM+03 Cadez, I., Heckerman, D., Meek, C., Smyth, P. ja White, S., Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7,4(2003), sivut 399–424.

CMS97 Cooley, R., Mobasher, B. ja Srivastava, J., Grouping web page references into transactions for mining world wide web browsing patterns. *KDEX '97: Proc. of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop*, Washington, DC, USA, 1997, IEEE Computer Society, sivu 2.

DvDD07 Doerr, C., von Dincklage, D. ja Diwan, A., Simplifying web traversals by recognizing behavior patterns. *HT '07: Proc. of the 18th Conf. on Hypertext and Hypermedia*, New York, USA, 2007, ACM, sivut 105–114.

EL05 Ezeife, C. I. ja Lu, Y., Mining web log sequential patterns with position coded pre-order linked wap-tree. *Data Mining and Knowledge Discovery*, 10,1(2005), sivut 5–38.

EV07 Eirinaki, M. ja Vazirgiannis, M., Web site personalization based on link analysis and navigational patterns. *ACM Trans. Inter. Tech.*, 7,4(2007), sivu 21.

FL05 Facca, F. M. ja Lanzi, P. L., Mining interesting knowledge from weblogs: a survey. *Data Knowl. Eng.*, 53,3(2005), sivut 225–241.

HPY00 Han, J., Pei, J. ja Yin, Y., Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29,2(2000), sivut 1–12.

JPT02 Jespersen, S., Pedersen, T. B. ja Thorhauge, J., A hybrid approach to web usage mining. *DaWaK 2000: Proc. of the 4th International Conference on Data Warehousing and Knowledge Discovery*, London, UK, 2002, Springer-Verlag, sivut 73–82.

JPT03 Jespersen, S., Pedersen, T. B. ja Thorhauge, J., Evaluating the markov assumption for web usage mining. *WIDM '03: Proc. of the 5th ACM international workshop on Web information and data management*, New York, USA, 2003, ACM, sivut 82–89.

LLY07 Ling, H., Liu, Y. ja Yang, S., An ant colony model for dynamic mining of users interest navigation patterns. *ICCA 2007. IEEE International Conf. on Control and Automation*, sivut 281–283.

LW04 Liu, J.-G. ja Wu, W.-P., Web usage mining for electronic business applications. *Proc. of the Third ACM international Conference on Machine Learning and Cybernetics, August 26-29, 2004, Shanghai, China.* IEEE Computer Society, 2004, sivut 1314–1318.

MDLN01 Mobasher, B., Dai, H., Luo, T. ja Nakagawa, M., Effective personalization based on association rule discovery from web usage data. *WIDM '01: Proc. of the 3rd International Workshop on Web Information and Data Management*, New York, USA, 2001, ACM, sivut 9–15.

MPTM08 Masseglia, F., Poncelet, P., Teisseire, M. ja Marascu, A., Web usage mining: extracting unexpected periods from web logs. *Data Mining and Knowledge Discovery*, 16,1(2008), sivut 39–65.

PHMAZ00 Pei, J., Han, J., Mortazavi-Asl, B. ja Zhu, H., Mining access patterns efficiently from web logs. *PADKK '00: Proc. of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, London, UK, 2000, Springer-Verlag, sivut 396–407.

RZP07 Ren, J., Zhang, X. ja Peng, H., On mining dynamic web clickstreams for frequent traversal sequences. *CIDM 2007. IEEE Symposium on Computational Intelligence and Data Mining*, sivut 477–484.

Sar00 Sarukkai, R. R., Link prediction and path analysis using markov chains. *Proc. of the 9th international World Wide Web conference on Computer networks : The international journal of computer and telecommunications netowrking*, Amsterdam, The Netherlands, 2000, North-Holland Publishing Co., sivut 377–386.

SLBR07 Stamey, J., Lassez, J.-L., Boorn, D. ja Rossi, R., Client-side dynamic metadata in web 2.0. *SIGDOC '07: Proc. of the 25th Annual ACM International Conf. on Design of Communication*, New York, USA, 2007, ACM, sivut 155–161.

TK02 Tan, P.-N. ja Kumar, V., Discovery of web robot sessions based on their navigational patterns. *Data Mining and Knowledge Discovery*, 6,1(2002), sivut 9–35.

TKK05 Ting, I.-H., Kimble, C. ja Kudenko, D., Ubb mining: finding unexpected browsing behaviour in clickstream data to improve a web site's design. *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*, sivut 179–185.

ZHF06 Zhou, B., Hui, S. C. ja Fong, A. C. M., Efficient sequential access pattern mining for web recommendations. *Int. J. Know.-Based Intell. Eng. Syst.*, 10,2(2006), sivut 155–168.