[6.2 Hypertekstin tiedonhakumalleja (jatkoa)]

3º Linkkitekstin huomiointi

[Chakrabarti, S. et al., Automatic resource compilation by analyzing hyperli
structure and associative text.
Computer Networks and ISDN Systems 30(1998), 65-74.]

ARC = Automatic Resource Compiler
- lähtökohdat:
Kleinbergin idea (arvo- ja napasivut)
manuaalisesti (ainakin osittain) tuotetut luokittelut (Yahoo, Infoseek);
voidaanko samaan tasoon päästä automaattisesti?

- perusidea tässäkin: linkin olemassaolo sisältää piilevää ihmisen tekemää sisällön kuvausta

ARC – algoritmi:

1. haetaan pieni joukko sivuja, kasvatetaan joukkoa linkkiyhteyksien per
2. painotetaan sivujen väliset linkit
3. valitaan tulossivut iteroimalla

1. lähes Kleinbergin perusmenetelmä:
hakukoneena AltaVista: 200 sivun juurijoukko
kasvatus perusjoukoksi: kaksi linkkiaskelta eli mukaan kaikki solmut jotka ovat yhden tai kahden linkin etäisyydellä
(tulos x00-3000 sivua)

2. painotusvaiheessa pyritään ottamaan mukaan tieto linkin ja aiheen kuvauksen ('kyselyn') yhteydestä:
- sivulla p oleva teksti <a href= .....to_page_q .... kuvaa sivun q sisältöä
- kuvauksen vahvuus määrää painon w(p,q) linkille (p,q) (p → q)

- määritellään B merkin kokoinen ankkuri-ikkuna linkkiankkurin yhteytee ainakin <a>-tagien välinen teksti

- w(p,q) = 1 + n(t), missä
n(t) = aiheen kuvauksen termien ja ankkuri-ikkunan tekstin osumien summa

- sopivaksi ikkunan koko on kokeiden mukaan esimerkiksi 50 sanaa

etäisyys	-100	-75	-50	-25	0	25	50	75	100
esiintymiä	1	6	11	31	880	73	112	21	7

3. iterointi:
lasketaan vektorit **h** ja **a**:
h(i) = sivun i arvo napasivuna, a(i) = sivun i arvo arvosivuna

merkitään **W** = linkkipainot sisältävä matriisi,
Z = matriisin **W** transpositio

- aloitus: **h** = (1, 1, ... , 1)
- suoritetaan k kertaa operaatiot

a = **W h**, **h** = **Z a**

- k iteraation jälkeen valitaan suurimman h-arvon omaavat sivut napas ja suurimman a-arvon omaavat sivut arvosivuiksi
- valittujen määrä esim. 15, iteraatioita vain k = 5

Menetelmän konvergenssi (vakaaseen tilaan) perustuu ominaisarvojen teoriaan. Pienellä arvolla k saavutetaan riittävän vakaa tila suurimpien a- ja h-arvojen osalta.

Tuloksen koko perustuu käytännön tekijöihin
('näytettäväksi sopiva' määrä sivuja).

Vektorien arvot pyrkivät kasvamaan, normeeraus välillä mahdollista.

Menetelmän evaluointi:

- koekäyttäjiä; vertailussa ARC, Yahoo, Infoseek (’supernapoja’ ...)

- aiheita esim. alcoholism, amusement parks, architecture,
  bicycling, classical guitar, computer vision, gardening, ...

- arvion kohteena saanti, tarkkuus (subjektiiviset),
  yleinen arvo (esim. selauksen aloitussivuna) (1-10)

- tulokset vaihtelevat paljon aiheittain
- ARC-sivut keskimäärin Infoseek-sivujen veroisia,
  vain hieman huonompia kuin Yahoo-sivut (n. 10 %)

Huom. ARC-tulosten esitystapa heikompi kuin muiden (tasoitusta ...)
ei yhteenvetokuvausta
ei ympäristön (hierarkian) kuvausta

4º PageRank-algoritmi (Google)

[Page, Brin et al., 1998 ...]

- linkki-informaation hyväksikäyttö (tässäkin ...); tavoitteena
  ”an approximation for the overall relative importance of web pages”

- suuri ’PageRank’ ilmaisee sivun tärkeyttä
- käyttö: hakujen tukena, selauksissa, verkkoliikenteen arvioinnissa

Intuitiivisia taustoja:
- PageRank kuvaa ’satunnaisen surfaajan’ todennäköisyyttä osua sivulle
  linkkejä seuraamalla; joskus hän hakee linkkien sijasta uuden
  ’satunnaisen’ sivun
  (syvällisempääkin teoreettista yhteyttä: random walk, stokastinen pros

- suuri PageRank <-> sivulle osoittaa monta linkkiä, tai
  sivulle osoittavat linkit tulevat suuren PageRank-arvon sivuilta
  (esim. tunnetusta hakemistosta, Yahoo tms.)

-yksinkertainen tulevien linkkien laskeminen ei riitä
vrt. navat, arvosivut; toiset linkit arvokkaampia kuin toiset

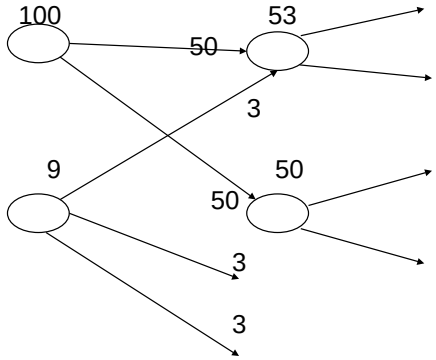
PageRank:

- sivun p osoittamat sivut: F(p)
- sivulle p osoittavat sivut: B(p)
- merkitään $N_u = |F(u)|$

- PageRank – perusmuoto: $$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

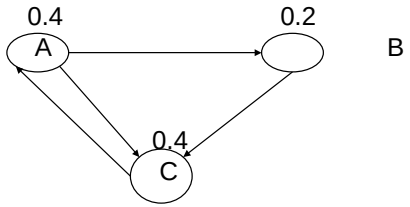
-sivun PageRank siis ’jaetaan’ siltä lähtevien linkkien osoittamille sivuille,
normeeraustekijällä c korjattuna
$c < 1$; kaikilla sivuilla ei ole seuraajia

Esim.



- määritelmä on rekursiivinen, mutta laskenta voidaan tehdä iteroimalla

-tasapainotilassa esim.



Tarkennettu määritelmä $R(u) = (1-c) + c\sum_{v \in B_u} \frac{R(v)}{N_v}$

Käytetty yleensä arvoa c = 0.85.

Algoritmin toteutus:

- Alkuarvot tarvitaan
  eivät periaatteessa vaikuta tulokseen, mutta konvergenssin nopeuteen
  (täsmällisiä arvoja/periaatteita ei kerrota (?))
  - iteraatioiden määrä ilmeisen suuri
    (noin 50; 322 milj. linkkiä, v. 1998 alussa)
    skaalautuu kuitenkin logaritmisesti

- Linkittömät sivut (’dangling links’) poistetaan ennen algoritmin käyttöä.

6.3 WebGlimpse: haun ja selauksen yhdistäminen

[Manber et al., 1996-97]

Hypertekstissä selaus yhtä normaalia kuin haku.

WebGlimpse: molemmat samassa järjestelmässä
- selaus tavanomaista
- haku voidaan kohdistaa halutun sivun
  ympäristöön (tai tehdä globaalisti)

(Glimpse: hakemistopohjalta,
WebGlimpse: WWW (+ paikalliset hakemistot)

Esim. find “network research” laitoksen materiaalista

1) kotisivun kautta
“network” tarkkuus huono
liian paljon selattavaa
tuloksen järjestys mielekäs ?
onnistuuko ‘refine search’?

“network research” saanti huono, koska
research ei ehkä esiinny tutkimussivuilla

2) tutkimusosaston sivuilta

“network” tuottaa hyvän tuloksen

- samoin tutkimukseen muuten liittyviltä sivuilta;
  haun konteksti tulisi voida muodostaa vähitellen,
  esim. selaamalla

WebGlimpse-ideat:

Glimpsen tapaan indeksointi ennen hakua:

1) sivujoukon analysointi
   lähtökohtana URL-joukko, siitä tehdään
   verkko paikallisia ja mahdollisesti muita
   linkkejä käyttäen
   - haluttu linkkietäisyys
     (erikseen paikallisille/muille linkeille)

2) ei-paikalliset sivut tallennetaan omaan
   hakemistoonsa

- muita täydennyksiä:
  kirjanmerkit, suosikit
  muita erillisiä, jopa 'historia' kontekstin luomiseksi

- indeksit voivat olla erilaisia (medium/small)

3) lasketaan sivujen ympäristöt (neighborhoods)

- osittain vaiheen 1 yhteydessä; eri tapoja
  linkkietäisyys
  kaikki sivun alihakemistot
  käyttäjän skriptinä määrittelemät sivut

  mahdollisesti useitakin ympäristöjä / sivu
  ainakin hierarkkisesti: suppea .. laaja
  yhteys luokitteluun tai ryvästykseen (?)

Esim. "tutkimussivut www.cs.helsinki.fi:ssä"

4) valituille sivuille lisätään hakukenttä, "search box"

- kohteen valinta: globaali / ympäristö
- yksinkertainen haku
- vaihtoehto: linkki erilliselle hakusivulle
  (advanced)

Haku Glimpse-ohjelmalla:

- likimääräinen haku, sana/merkkijono, case (a/A)
- tuloksen koon rajoittaminen
- vain tiettynä aikana päivitetyt sivut

Haun tulos:
- sivun otsikko, (linkki,) kontekstia
- osuman rivinumero, rivin sisältö
- osumien korostettu näyttö
- mahdollisesti siirtyminen suoraan osumariville
- sivun päivitysaika

Kokemuksia:
- 3 linkin etäisyys tuottaa kovin suuria
  ympäristöjä; paljon linkkejä
- indeksointi voi olla hidasta
  (esim. 150 MB, 3 h;
  ympäristöjen ja hakukenttien luonti)
- haku käytännössä yhtä nopea globaalisti tai
  ympäristöstä (indeksoinnin jälkeen)

- lisäpiirteitä:
  henkilö- tai aihekohtaiset kokoelmat
  yhteys Scatter/Gather-tekniikkaan
  selaushistorian ylläpito (ympäristöineen)

6.4 Yleistä selauksen ja haun suhteesta:

1) Haun ongelmia:
kyselyn tekemisen vaikeus;
tiedontarpeen selkeys
kyselyn muoto
(kuinka saada tulokseen vain relevantteja ...)

haun suoritustapa voi olla käyttäjälle tuntematon
(termiesiintymien merkitys, mitä muuta?)

haun konteksti ei yleensä vaikuta; käyttäjä tuntee sen

2) Selauksen ongelmia:

hidasta
vaikeaa pysyä selvillä materiaalin organisoinnista
(mikä olisi selauksessa tärkeää)
vaikeaa pysyä olennaisessa
(ylimääräisen näkemisestä voi olla etuakin)

Yhdistelmiä:

- haku luokitteluun perustuen (luokasta)

- selauksen yhteydessä ehkä ‘find related/similar sites’
(samanlaisuuden määrityksen ongelmat)

(s. 10) - 1) aloitussivun haku, 2) selaus tässä ympäristössä, tai

- 1) löydetään selaamalla jokin kohde (tai useita),
2) halutaan hakea kohteesta ja ympäristöstä