Webes adatbányászat

Készítette: Göbölös-Szabó Julianna

2010. május 25.

Bevezetés

- Webgráf: csúcsok a weblapok, élek az oldalak közti hiperlinkek
- Webgráf mérete: kb 200 millió bejegyzett domain (ebből kb 85 millió aktív) $\longrightarrow$ hatékony algoritmus kell
- Fontos feladatok:
  1. Weboldalak rangsorolása (PageRank, Persoalized PageRank)
  2. Két lap közti hasonlóság mérése (SimRank, Jaccard együttható)

Webes keresés története

- Az internet létrejöttével felmerült a fájlok közt való keresés igénye. Eleinte csak a fájlnévben lehetett keresni (pl.90-es években Archie és Veronica)
- Később megjelent a tartalom alapú indexelés, pl. Aliweb. A felhasználó leírást készített a dokumentumról, és ez alapján történt a keresés. Nagyon pontos keresés, de szűk adattéren.
- Mai napig az automatikus indexelés a legelterjedtebb, ennek előfeltétele, hogy a tárolt dokumentumok gyorsan elérhetők legyenek

A webes keresés követelményei

- Érteni kell a felhasználó szándékát
- Releváns választ kell generálni
- Fontos a rangsorolás jósága, a találatok megjelenítése

A webes keresés követelményei

- Érteni kell a felhasználó szándékát
- Releváns választ kell generálni
- Fontos a rangsorolás jósága, a találatok megjelenítése
- **Követelmények:**
  1. Széleskörűség
  2. Naprakészség
  3. Rangsorolás
  4. Megjelenítés

A webes dokumentumok feldolgozása:

- A dokumentum feldarabolása szavakra, operátorokra
- Szavak átalakítása belső szóazonosítóvá
- A dokumentumok hozzárendelése belső szóazonosítókhoz
- A szóhoz tartozó előfordulásokat nyilvántartó invertált index elkészítése
- A szóhoz tartozó metaadatok kigyűjtése
- Kapcsolati indexek létrehozása

Keresőkifejezésre illeszkedő dokumentumok meghatározása

- Keresőkifejezés elemzése, felbontása szavakra
- Szavak konvertálása a megfelelő nyelvtani alakra
- Illeszkedő dokumentumok meghatározása az invertált lista alapján
- Az illeszkedő dokumentumok rangsorba állítása
- Találati lista limitálása
- Limitált találati lista visszaküldése

A PageRank alapötlete

- Brin-Page algoritmus, 1998.
- Egy oldal fontos, ha fontos oldalak mutatnak rá.
- N db weboldal van, köztük linkek futnak, a struktúrát $A_{N\times N}$ mátrix írja le:

$$a_{ij} = \begin{cases} \frac{1}{n} & \text{ha van } i \to j \text{ link, és } n \text{ link található az } i. \text{ oldalon,} \\ 0 & \text{egyébként.} \end{cases}$$

- **Tétel** Legyen $A_{N\times N}$ sorsztochasztikus mátrix, $\underline{j} = (\frac{1}{N}, \ldots, \frac{1}{N})$. Ekkor $\underline{p} = \lim_{m\to\infty} \underline{j}A^m$ létezik és $\underline{p}A = \underline{p}$.
- (Az ilyen $\underline{p}$ vektort a lapok rangvektorának hívjuk.)

A PageRank algoritmus

Algoritmus:

1. Inicializálás:
   - Készítsük el A mátrixot!
   - Legyen $\underline{p}^0 = (\frac{1}{N}, \ldots, \frac{1}{N})$!
2. Iteráció: $\underline{p}^{i+1} = \underline{p}^i A$
3. Leállási feltétel: Ha $|\underline{p}|$ már alig változik, vagy ha a $\underline{p}$ által meghatározott sorrend már nem sokat változik.

Szemlélet: sztochasztikus szörfölő

Az igazi PageRank

- Az előző algoritmus könnyen kijátszható.
  1. **Zsákutca** probléma: Ha létezik olyan csúcs, amiből nem mutat ki él.
  2. **Pókháló** probléma: néhány lap csak egymásra mutat.

Az igazi PageRank

- Az előző algoritmus könnyen kijátszható.
  1. **Zsákutca** probléma: Ha létezik olyan csúcs, amiből nem mutat ki él.
  2. **Pókháló** probléma: néhány lap csak egymásra mutat.
- **Ötlet:** lapok „megadóztatása”, azaz A vektor helyett használjuk:

$$B = \varepsilon \cdot U + (1 - \varepsilon) \cdot A$$

  ahol U mátrixban minden $u_{ij} = \frac{1}{N}$.
- **Szemlélet:** szeszélyes sztochasztikus szörfölő.
- Tapasztalat: kb 52 iterációval elérhető a leállási feltétel

Personalized PageRank motivációja

- A PageRank a fontosságot „demokratikusan” határozza meg
- De az egyes felhasználóknak saját preferenciái lehetnek, némely oldalakat gyakrabban nézik, mint a többit
- PageRanket indítsuk a felhasználó által preferált oldalról
- Témaérzékenyebb keresést tesz lehetővé

Personalized PageRank

- $\underline{p}$ rangvektort az alábbi egyenlet megoldásaként kapjuk:

$$\underline{p} = (1 - c) \cdot \underline{p}A + c \cdot \underline{r}$$

- $\underline{r} = (\frac{1}{N}, \ldots, \frac{1}{N})$ választással az eredeti PageRanket kapjuk
- **Tétel** Bármely $\underline{r}_1$, $\underline{r}_2$ preferenciavektorokra és olyan α_1, $\alpha_2 \geq 0$ konstansokra, melyekre $\alpha_1 + \alpha_2 = 1$ teljesül:

$$PPV(\alpha_1 \underline{r}_1 + \alpha_2 \underline{r}_2) = \alpha_1 PPV(\underline{r}_1) + \alpha_2 PPV(\underline{r}_2)$$

(Azaz a PPR érték lineáris függvény.)

Probléma a számításigénnyel

- Minden u oldalhoz ki kellene számítani a $PPV(u)$ vektort, ez még offline számítás esetén is túl sok időt igényel ($O(l \cdot N^2)$ lépés)
- **Skálázható algoritmus** kell:
  1. Az index adatbázis számolása egy rendező algoritmusnak megefelelő nagyságrendű legyen
  2. Egy lekérdezéshez elég legyen konstans sok adatbázis hozzáférés
  3. Az algoritmus futtatható legyen külső memóriából (a webgráf nem fér el a memóriában)
  4. Párhuzamosítható legyen

Monte Carlo-módszer

- Olyan sztochasztikus szimulációs módszer, amely számítástechnikai eszközök segítségével előállítja egy adott kísérlet végeredményét
- Az eredményként kapott numerikus jellemzőket feljegyzik és kiértékelik.
- Az eredmény hibájának meghatározása szórás kiszámításával történik.

Personalized PageRank számítása Monte Carlo-módszerrel 1.

- **Tétel** Legyen L valószínűségi változó $P(L = i) = c \cdot (1 - c)^i$ eloszlással ($c \in (0, 1)$). Tekintsünk egy u pontból induló véletlen sétát, melynek hossza L. Ekkor a $\underline{p} = PPV(u)$ vektor v-dik koordinátája:

$$PPV(u, v) = \mathbf{P}\{\text{a véletlen séta } v\text{-ben végződik}\}$$

- **Fingerprint path:** u csúcsból induló L hosszú séta.
- **Fingerprint:** fingerprint path végpontja.
- Minden u csúcshoz (weboldalhoz) készítünk K független sétát (fingerprintet) készítünk, majd ezekből becsüljük $PPV(u)$ vektort.
- Gyakorlatban jó paraméterek: $N = 1000$, $L = 12$

Algoritmus

- **Inicializálás:** Minden u oldalhoz fenntartunk egy P tömböt, kezdetben N elemmel, mindegyik elem (u, u), $FingerPrint[u] = \emptyset$
- Amíg $P \neq \emptyset$: rendezzük a P-beli párokat, majd minden $(u, v) \in P$ párra :
  - $w := v$ egy véletlen ki-szomszédja
  - ha ($random < c$), akkor w-t tegyük be $Fingerprint[u]$-ba, és töröljük P-ből az aktuálisan vizsgált párt
  - különben (u, v)-t helyettesítsük (u, w)-vel

SimRank

- Oldalak közti hasonlóság számítására
- „Két oldal hasonló, ha hasonló oldalak hivatkoznak rájuk”

$$sim(u,v) = \frac{c}{|I(u)| \cdot |I(v)|} \cdot \sum_{i=1}^{|I(u)|} \sum_{j=1}^{|I(v)|} sim(I_i(u), I_j(v))$$

ahol $c \in (0,1)$ konstans, $I(x)$ az x csűcs be-szomszédainak halmaza ($|I(u)| = 0$ vagy $|I(v)| = 0$, akkor $sim(u,v) = 0$)

- Iterációval lehet számolni, de nem hatékony

$$sim_{l+1}(u,v) = \frac{c}{|I(u)| \cdot |I(v)|} \cdot \sum_{i=1}^{|I(u)|} \sum_{j=1}^{|I(v)|} sim_l(I_i(u), I_j(v))$$

SimRank számítása Monte Carlo módszerrel

- Fingerprinteket készítünk minden u csúcsból a be-éleken haladva
- Legyen $\tau_{u,v}$ az első időpillanat, amikor az u-ból és v-ből indult séták találkoznak és $\tau_{u,v} = \infty$, ha sosem találkoznak.
- **Tétel** Független, (be-linkeken) visszafele haladó l hosszú sétákra: $sim_l(u, v) = \mathbf{E}[c^{\tau_{u,v}}]$
- Elegendő K db független sétát generálni, és ezekben kapott sim értékek átlagát tekinteni

Fingerprint tree

- Ha két séta egyszer találkozik, onnantól ők együtt mennek tovább

  Fingerprint tree konstrukciója
- Csúcsok a weboldalaknak felelnek meg (értékeik $1, 2, \ldots, N$)
- $(u \longrightarrow v)$ élet behúzzuk $\tau_{u,v}$ élsúllyal, ha
  1. $v < u$ és u és v találkoznak $\tau_{u,v}$-ben
  2. az 1.-t teljesítő csúcsok közül v-hez tartozik a legkisebb $\tau_{u,v}$
  3. az 1.-t és 2-t teljesítő csúcsok közül v indexe minimális
- Az így kapott erdőben minden csúcsnak legfeljebb 1 ki-szomszédja van
- Bármely u, v csúcsra $\tau(u, v)$ egyszerűen megkapható az FPT alapján

FPT konstrukciója

- K fingerprintet készítünk a Monte Carlo módszerhez
  Egy iteráció:
  for $i = 1$ to l
  1. Minden v csúcshoz generálunk egy $NextIn[v]$ csúcsot
  2. Minden u csúcsra, amire $PathEnd[u] \neq$ stopped, $PathEnd[u] = NextIn[PathEnd[u]]$
  3. FPT frissítése
  4. Ha vannak találkozó utak, akkor a nagyobbik j indexre $PathEnd[j] =$ stopped

P-SimRank

- A SimRank esetén előfordulhat, hogy két népszerű lapra ugyanazok az oldalak mutatnak, mégis az előző módszer rossz eredményt ad

P-SimRank

- A SimRank esetén előfordulhat, hogy két népszerű lapra ugyanazok az oldalak mutatnak, mégis az előző módszer rossz eredményt ad
- Módosítás:

$$psim_{l+1}(u,v) = c \cdot \left(\frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|} \cdot 1 + \right.$$

$$+ \frac{|I(u) \setminus I(v)|}{|I(u) \cup I(v)|} \cdot \frac{1}{|I(u) \setminus I(v)| \cdot |I(v)|} \sum_{u' \in I(u) \setminus I(v)} \sum_{v' \in I(v)} psim_l(u', v') +$$

$$\left. + \frac{|I(v) \setminus I(u)|}{|I(u) \cup I(v)|} \cdot \frac{1}{|I(v) \setminus I(u)| \cdot |I(u)|} \sum_{v' \in I(v) \setminus I(u)} \sum_{u' \in I(u)} psim_l(u', v') \right)$$

Jaccard-együttható

- Hasonlóságot mér egy lépésben:

$$Jac(u,v) = \frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$$

- Kiterjeszthető több lépésre
- k-távolságra levő szomszédokra nézzük a Jaccard-együtthatót
- Exponenciális súllyal súlyozzuk a távolabbi szomszédokat, azaz:

$$XJac_l(u,v) = \sum_{k=1}^{l} \frac{|I_k(u) \cap I_k(v)|}{|I_k(u) \cup I_k(v)|} \cdot c^k \cdot (1-c)$$

XJac számolása Monte Carlo-módszerrel

- **Algoritmus** - egy fingerprint számolása
  1. Generáljunk egy véletlen σ permutációt
  2. Minden j csúcsra $NFP[j] = \sigma(j)$
  3. for $k = 1$ to l
     - $FP[] = NFP[]$
     - Minden (u, v) élre $NFP[v] = \min \quad \{NFP[v], FP[u]\}$
     - Mentsük el $NFP[]$ tömböt FP_k-ként
  4. Egyesítsük FP_k tömböket, és készítsük el az invertált indexet
- Egy iteráció után annak a valószínűsége, hogy $FP[u] = FP[v]$, $\frac{|I_k(u) \cap I_k(v)|}{|I_k(u) \cup I_k(v)|}$ lesz
- $N = 100$ és $l = 4$ paraméterekkel jó eredményt kapunk

Felhasznált irodalom

- Tikk Domonkos: Szövegbányászat
- Bodon Ferenc: Adatbányászat
- Fogaras Dániel, Rácz Balázs, Csalogány Gábor, Sarlós Tamás: Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds and Experiments
- Fogaras Dániel, Rácz Balázs: Scaling Link-Based Similarity Search