Webanalízis, avagy melyik a legértékésebb csúcs egy gráfban?

Mátyás Péter

2011. május 16.

1. Bevezetés

A World Wide Web megszületése óta eltelt 20 év alatt az Internet az emberek mindennapjainak részévé vált. A kezdetben csak pár oldalból álló hálózat 2009. márciusában már több mint 25 milliárd [1], napjainkban pedig 35 milliárd [2] egymással hiperhivatkozásokon keresztül összekötött oldalt tartalmaz a legváltozatosabb témákról szolgáltatva információt. Manapság egy átlagos felhasználó az Internet elé ülve a rutinszerű tevékenységei (mint például a postafiók és pár gyakran látogatott oldal meglátogatasa) után vagy véletlenszerűen böngészi tovább a világhalót, vagy célirányosan próbál olyan oldalakat meglátogatni, ahol számára releváns információk lehetnek.

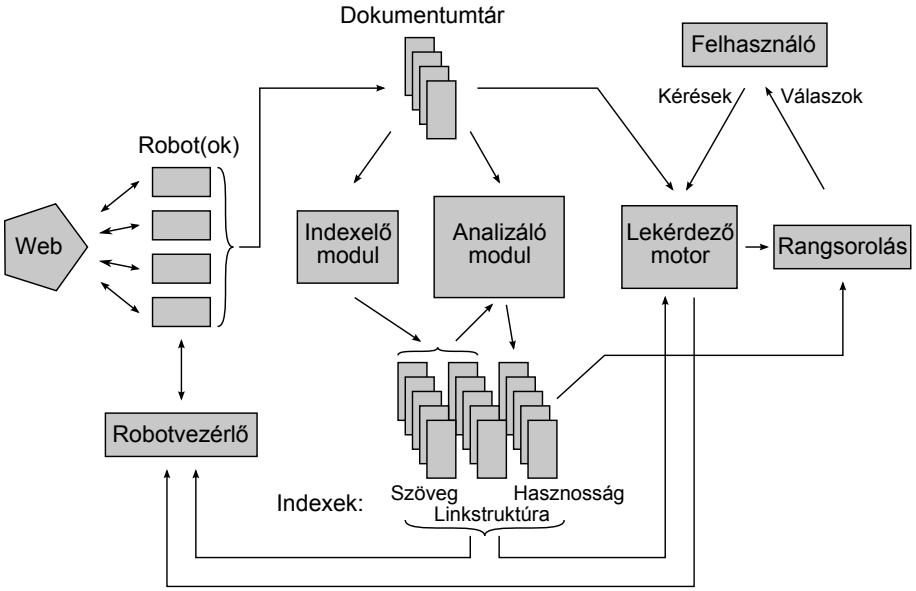
Tekintettel a World Wide Web nagyságára, a hasznosnak bizonyuló oldalak megtalálása csupán bolyongással nem célravezető. A probléma megoldására hozták létre a keresőket. A felhasználók megadnak a keresőnek egy kérést, ami tipikusan kulcsszavak listájából áll, amire válaszként weboldalak listáját kapják. Általában ezek az oldalak tartalmazzák a keresőkifejezéseket is, de csupán ennyi nem elegendő, hiszen sok ezer, de akár millió oldal is tartalmazhatja ezen kifejezéseket. Ezért fontos, hogy a találati listát valamilyen módon relevancia szerint rangsoroljuk, hiszen a legtöbb felhasználó csak a tlista első oldalán – annak is az elején – felsorolt oldalakat tekinti meg. A 90-es évek vége felé használatos 4 legelterjedtebb keresőből még csak 1 találta meg saját magát, azaz csak 1-

nek sikerült a saját találati listáján a top 10-be kerülnie [5], a mai rangsoroló algoritmusoknál ilyen probléma már nem lép fel.

Dolgozatom célja röviden felvázolni, hogyan is néz ki egy keresőrendszer, majd pedig bemutatni egy, a web gráfszerkezetén alapuló rangsoroló algoritmust.

2. Webes keresőrendszerek

A következőkben tekintsük át egy keresőrendszer általános felépítését [3], melyek a az 1. ábrán látható módon épülnek fel. Legfontosabb részei a weboldalak letöltését végző *robot* és az őt szabályozó *robotvezérlő*, a begyűjtött dokumentumokat tároló *dokumentumtár*, a kulcsszavas keresés alapját szolgáló *indexelő modul*, az *analizáló modul* és a *rangsorolás*.



1. ábra. Egy keresőrendszer általános felépítése

A robotok olyan programok, amelyek egy – sok hivatkozást tartalmazó – oldalról kiindulva böngészik a webet, a meglátogatott oldalakat letöltik és a *dokumentumtárban* tárolják el. Mivel a meglátogatandó weboldalak száma nagyon nagy, ezért belátható időn belül nem lehet minden oldalt letölteni, nem is beszélve a szükséges tárolókapacitásról. Ezért egy URL-eket tartalmazó listából indul ki. Meglátogatja a listán elsőnek szereplő oldalt, kigyűjti belőle a linkeket, majd

átadja a *robotvezérlő* modulnak. Ez a modul dönti el, hogy mely oldalakat kell a későbbiekben letölteni illetve melyek azok, amelyek lényegtelenek, esetleg már szerepeltek a letöltött oldalak listáján. Ahhoz, hogy a robotvezérlők eldönthessék, mely oldalakat érdemes letölteni, az oldalakhoz különböző fontossági mérőszámokat rendelnek [3].

A *dokumentumtárban* történik meg az adatok előfeldolgozása a keresőrendszer többi része számára. Ha egy oldal megváltozik, akkor az adatbázisból törölni kell a régi példányát és helyettesíteni kell azt az újjal. Megállapították, hogy az oldalaknak 40 százaléka hetente változik, sőt a .com domain-űeknek csaknem ötöde napi szinten frissül [9]. Ezért a robotoknak óvatosan kell dönteniük az újra meglátogatandó helyeket illetően, hiszen ettől függ majd, mennyire naprakész a gyűjteményünk. Az indexelő modul kigyűjti az összes előforduló szót az adatbázisban szereplő oldalakról és elkésziti au őket tartalmazó URL-ek listáját (kivételek a „tiltólistás szavak, pl.: névelők). Amikor rákeresünk egy szóra, akkor a *szöveg index* jelöli ki azokat az oldalakat, amelyek a végső találati listába kerülhetnek. Többszavas keresés esetén a találathalmazok metszete lesz a lehetséges oldalak listája.

A *lekérdező motor* felelős a felhasználóktól érkező kérések kezeléséért. Az első keresők csupán az oldalakon található szöveges információt használták fel a rangsoroláshoz. A kiinduló ötlet az volt, hogy az az oldal kerüljön előkelőbb helyre a listán, melyben a keresett szó többször szerepel [12]. Ezt a rangsorolást érdemes kombinálni a webgráf linkstruktúráján alapú további algoritmusokkal.A legjobb minőségű rangsoroló eljárások a szöveg és a hivatkozások igen nagy számú tényezőit veszik figyelembe, amelyeket gyakran a gépi tanulás eszközeivel kombinálnak [6].

3. A PageRank algoritmus

A gráf alapú rangsorolás a weboldalak közti linkeken alapul. Feltételezi, hogy egy oldal akkor tartalmaz hiperhivatkozást egy másikra, ha azt a hivatkozás elhelyezője jónak tartja. Magyarul az ott található információ hasznosnak bizonyul számára. Bár az ilyen véleménnyilvánítás eléggé szubjektív, és előfordulhat, hogy a belinkelt oldal nem is annyira hasznos [7], illetve vannak nem linkelt, de sokkal

hasznosabb helyek is weben, ennek ellenére a hiperlinkek nagyon jó alapot szolgálnak a rangsoroláshoz. Főleg, ha abba is belegondolunk, hogy az imént említett szubjektivitás okozta eltérések nagy számú oldalnál illetve hivatkozásnál kiátlagolódnak. A World Wide Web egy olyan óriási gráf, ahol az előzőek nyilván teljesülnek.

A PageRank algoritmus [8] az elsők közé tartozott, mely a weboldalak közti linkrendszert kihasználva készítette el a rangsorát. Az feltételezés, miszerint egy oldalra akkor hivatkozik valaki, ha azt jónak tartja az illető, helyesnek bizonyult, hiszen az ezen az ötleten alapuló Google kereső óriási sikernek örvendett és örvend ma is.

Az algoritmus több, mint a csúcsok befokainak meghatározása, ez ugyanis nagyon könnyen manipulálható [10]. Figyelembe veszi azt is, hogy az adott oldalra mutató linkek mennyire fontos oldalról indulnak ki.

Mindez jellemezhető egy *sztochasztikus szörfölő* gráfon való véletlen sétájával [11]. A szörfölő egyenletes eloszlás szerint választ kiindulópontot a gráf n csúcsa közül, majd arról a csúcsról, ahol éppen tartózkodik, szintén egyenletes eloszlás szerint ugrik valamelyik szomszédjára. Sok idő (lépés) után közelítőleg p_i valószínűséggel lesz az i csúcson [8].

Ez a modell nem tudja jól kezelni a gráf azon csúcsait, melyekből nem vezet ki él, vagyis melyeknek a kifoka 0. A *szeszélyes sztochasztikus szörfölő* ezzel szemben minden lépés után d valószínűséggel megszakítja az útját és egyenletes eloszlás szerint kijelölt véletlen helyre ugrik a gráfban. A többi esetben ugyanúgy viselkedik, mint a sztochasztikus szörfölő (ennek $(d-1)$ a valószínűsége). A d értéket tipikusan $0,1$ és $0,2$ között szokás megválasztani.

3.1. Véletlen bolyongás a Web gráfján

A PageRank előkészítése képpen az elugrás nélküli folyamatot írjuk le, amely egy véletlen bolyongás a web gráfján. Tekintsünk egy n csúcsú irányított gráfot. A gráf egy j csúcsába mutató éleinek kezdőpontjai alkotják a $B(j)$ halmazt: $B(j) = \{k : k \to j\}$. Az algoritmus a gráf minden egyes csúcsához egy súlyt

rendel, melyek egy n komponensű $\mathbf{p}$ vektort határoznak meg:

$$p_i = \sum_{j \in B(j)} \frac{p_j}{|B(j)|}. \tag{1}$$

Algoritmus szempontjából az egyes p_i értékeket iterációval kapjuk meg, ahol az alábbi rekurzív formulát használjuk:

$$\forall i \, [p_i]_1 := \frac{1}{n} \tag{2}$$

$$\forall i \, [p_i]_k := \sum_{j \in B(j)} \frac{[p_j]_{k-1}}{|B(j)|}. \tag{3}$$

A fenti gondolatokat másképp is meg lehet fogalmazni. Gráfunk, melyet a továbbiakban G-vel jelölünk álljon n csúcsból ($|G| = n$). Definiáljunk egy $n \times n$-es $\mathbf{A} = (a_{ij})$ átmeneti mátrixot, melynek sorait és oszlopait G elemeivel indexeljük. Tegyük fel, hogy az $i \in G$ csúcsból a G gráf d_i darab csúcsához vezet él. Ekkor $d_i \geq 1$ minden i-re. Ezek után legyen

$$a_{ij} = \begin{cases} \frac{1}{d_i} & \text{ha } i \to j \text{ él benne van } G\text{-ben,} \\ 0 & \text{különben.} \end{cases} \tag{4}$$

Ekkor az $\mathbf{A}$ mátrix sorsztochasztikus, tehát egy G csúcshalmazú Markov-lánc átmeneti mátrixának tekinthető.

Legyen $\mathbf{u} = (\frac{1}{n}, \ldots, \frac{1}{n})$ egy n komponensű sorvektor. Ha feltesszük, hogy $\mathbf{p} = \lim_{t \to \infty} \mathbf{u}\mathbf{A}^t$ létezik, akkor $\mathbf{p}$ stacionárius eloszlása lesz $\mathbf{A}$-nak, azaz $\mathbf{p}\mathbf{A} = \mathbf{p}$.

Sajnos, amikor a gráf nem aperiodikus ill. nem erősen összefüggő, olyankor a $\mathbf{p} = \lim_{t \to \infty} \mathbf{u}\mathbf{A}^t$ határérték nem létezik. További probléma, hogy a fenti módszerrel meghatározott p_i értékek könnyedén manipulálhatók [4]. Ha azonban az algoritmusban bevezetjük az elugrást is, vagyis áttérünk a szeszélyes sztochasztikus szörfölő modelljére, akkor a fenti problémák megszüntethetők.

3.2. PageRank

Egy d valószínűségű elugrás bevezetésével (1) mintájára a PageRank érték:

$$p_i = (1-d) \sum_{j \in B(j)} \frac{p_j}{|B(j)|} + \frac{d}{n}, \tag{5}$$

Ez alapján az iterációs képlet:

$$\forall i \, [p_i]_1 := \frac{1}{n} \tag{6}$$

$$\forall i \, [p_i]_k := (1-d) \sum_{j \in B(j)} \frac{[p_j]_{k-1}}{|B(j)|} + \frac{d}{n}. \tag{7}$$

Másképp ez úgy is fogalmazható meg, hogy az eddigi $\mathbf{A}$ mátrixunk helyett egy

$$\mathbf{B} := (1-d) \cdot \mathbf{A} + d \cdot \mathbf{U} \tag{8}$$

írja le a sétát, ahol az $n \times n$-es $\mathbf{U}$ mátrix minden sora egy n elemű egyenletes eloszlás $\mathbf{u} = \left(\frac{1}{n}, \cdots, \frac{1}{n}\right)$ vektora. Vagyis:

$$\mathbf{U} = \begin{pmatrix} \mathbf{u} \\ \mathbf{u} \\ \vdots \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}. \tag{9}$$

Mivel az így generált $\mathbf{B}$ mátrix is sorsztochasztikus, ezért ez egy Markov-lánc mátrixa, amely jól leírja a szeszélyes szörfölőt.

3.3. A stacionárius eloszlás létezése

Tekintsük π_i-t az i csúcs fontosságának. A $\mathbf{p}$ vektor pontos értékét nem kell ismernünk, ugyanis a rangoroláshoz elegendő a komponenseket összehasonlítani. Az $\mathbf{uB}^t$ vektorok iterációval számolhatók az $\mathbf{uB}^t = \mathbf{uB}^{t-1}\mathbf{B}$ azonosság alapján. Az 1. **tétel** bizonyításából következik, hogy a konvergencia legalább olyan gyors, mint az $(1-d)$ hányadosú mértani soré. Ezért elegendő az első pár iterációs

lépésig elmenni. A következőkben bebizonyítjuk az $\mathbf{uB}^t$ sorozat konvergenciáját.

1. Tétel. *Legyen $0 < d < 1$ egy valós szám. Ekkor tetszőleges π kezdőeloszlás esetén a*

$$\pi((1-d)\mathbf{A} + d\mathbf{U})^t$$

sorozat konvergál egy csupa pozitív elemű $\mathbf{p}$ stacionárius eloszláshoz, ahogy t tart a végtelenhez. A $\mathbf{p}$ független a π kezdőeloszlástól.

Bizonyítás: A zárójeleket felbontva

$$\pi((1-d)\mathbf{A} + d\mathbf{U})^t = \sum c\mathbf{X}_t\mathbf{X}_{t-1}\cdots\mathbf{X}_2\mathbf{X}_1$$

adódik, ahol olyan t hosszúságú $\mathbf{X}_t\cdots\mathbf{X}_2\mathbf{X}_1$ szorzatokat összegzünk, melyek minden tényezője $\mathbf{A}$ vagy $\mathbf{U}$. A c együttható értéke $d^k(1-d)^{t-k}$, ha a sorozatban pontosan k darab $\mathbf{U}$ található.

Az eddigieket úgy szemléltethetjük, hogy egy $(d, 1-d)$-érmét dobunk fel t-szer (az érmedobások egymástól függetlenek). Ha az i-edik dobás fej, akkor $\mathbf{X}_i = \mathbf{U}$, ha írás, akkor pedig $\mathbf{X}_i = \mathbf{A}$. Az $\mathbf{X}_t\cdots\mathbf{X}_2\mathbf{X}_1$ együtthatója (c_j) éppen az előfordulásának valószínűségét adja majd meg. Mivel a pénzfeldobások geometriai eloszlást követnek, ezért

$$c_j = d(1-d)^j. \tag{10}$$

Az $\mathbf{A}$ mátrix sorsztochasztikus, ezért $\mathbf{AU} = \mathbf{UU} = \mathbf{U}$ teljesül. Ha tehát az $\mathbf{X}_t\cdots\mathbf{X}_2\mathbf{X}_1$ szorzatnál $j+1$ a legkisebb index, melyre $\mathbf{X}_{j+1} = \mathbf{U}$, akkor $\mathbf{X}_t\cdots\mathbf{X}_2\mathbf{X}_1 = \mathbf{UA}^j$. Ezt figyelmbe véve

$$((1-d)\mathbf{A} + d\mathbf{U})^t = (1-d)^t\mathbf{A}^t + \sum_{j=0}^{t-1} c_j\mathbf{UA}^j \tag{11}$$

alakba írható. Ezek alapján a $\pi\mathbf{U} = \mathbf{u}$ így írható:

$$\pi((1-d)\mathbf{A} + d\mathbf{U}))^t = \pi(1-d)^t\mathbf{A}^t + \sum_{j=0}^{t-1} d(1-d)^j\mathbf{jA}^j \tag{12}$$

Mivel $\pi\mathbf{A}^t$ egy eloszlásvektor, vagyis minden tagja legfeljebb 1, ezért (12) jobb-

oldalának első tagja nullvektorhoz tart. A $\pi((1-d)\mathbf{A}+d\mathbf{U}))^t$ tehát a

$$\sum_{j=0}^{\infty} d(1-d)^j \mathbf{u}\mathbf{A}^j\text{-hoz} \tag{13}$$

fog tartani. Ez a sorösszeg létezik, ugyanis komponensenként majorálható a $\sum_{j=0}^{\infty} d(1-d)^j$ sorral. Másrészt az összegvektor minden komponense legalább $\frac{d}{n}$ az első tag miatt. Ezzel igazoltuk a tétel állítását. ■

4. Webgráf és random gráf vizsgálata Mathematicával

4.1. Crawler

Első feladatként egy, a Mathematicával megírt Crawlert tanulmányoztunk. Ez azt tudta, hogy egy adott csúcsból (linkről) kiindulva bejárta annak szomszédjait egy adott mélységig. A teszteléshez készítettünk egy teszt weboldalt több linkkel. Azt tapasztaltuk, hogy a jól megírt weboldalakat ügyesen be tudta barangolni. Viszont egy teljesen véletlen oldalt beírva kezdőcsúcsnak, hibaüzeneteket kaptunk. Ez azzal magyarázható, hogy a bennük szereplő linkeket nem tudta követni.

Ez a feladat igazából csak szemlélteti, hogy mennyire összetett a World Wide Web linkstruktúrája. A továbbiakban véletlen gráfokat használtunk a vizsgálataink-hoz.

4.2. Véletlen bolyongás és PageRank

A következőkben a PageRank számítására alkalmas függvényt írtunk meg és ellenőriztük működését. A mellékelt *matyas_peter.nb* fájl tartalmazza ezen eredményeket. Azt tapasztaltuk, hogy a szeszélyes sztochasztikus szörfölő modellje valóban kielégíti a felteteleket. Olyankor ugyanis szemmel láthatóan konvergál a stacionárius eloszláshoz a PageRank vektor. Továbbá módszerünket igazolja az is, hogy a Mathematica beépített függvénye is ugyanazt az eredményt adta.

Hivatkozások

[1] Wikipedia www szócikk. `http://en.wikipedia.org/wiki/Www`, 2011. május 7., 15:03.

[2] A World Wide Web mérete. `http://www.worldwidewebsize.com`, 2011. május 7., 15:15.

[3] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the Web. *ACM Transactions on Internet Technology*, 1(1), 2001.

[4] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. Pagerank as a function of the damping factor. In *Proceedings of the 14th World Wide Web Conference (WWW)*, pages 557–566, 2005.

[5] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[6] O. Chapelle, Y. Chang, and T-Y Liu. The yahoo! learning to rank challenge, 2010.

[7] Brian D. Davison. Recognizing nepotistic links on the web. In *AAAI-2000 Workshop on Artificial Intelligence for Web Search, Austin, TX*, pages 23–28. Artificial Intelligence for Web Search, Technical Report WS-00-01, July 30 2000.

[8] D. Easley and J. Kleinberg. *Networks, Crowds and Markets*. Cambridge University Press, 2010.

[9] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet L. Wiener. A Large-Scale Study of the Evolution of Web Pages . *Software—Practice and Experience - Special issue: Web technologies*, 34(2), 2004.

[10] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *First International Workshop on Adversarial Information Retrieval on the Web*. Citeseer, 2005.

[11] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[12] Stephen E. Robertson and Karen Sparck Jones. Relevance weighting of search terms. In *Document retrieval systems*, pages 143–160. Taylor Graham Publishing, London, UK, UK, 1988.