PageRank algoritam s visećim vrhovima (*dangling nodes*)

Marina Matešić[1], Tomislav Novak[1], and Timotej Repak[1]

[1]Prirodoslovno-matematički fakultet

U ovom radu obrađujemo PageRank algoritam za rangiranje web stranica po važnosti. Posebna pozornost dana je visećim vrhovima. Nakon teorijske analize priložen je i kod programa.

1 Uvod

PageRank algoritam i metode koje koristi razvili su Sergey Brin i Larry Page, tadašnji studenti na Sveučilištu Stanford 1998. godine. Tad datiraju njihovi prvi radovi na tu temu, i izlaganje na 7. međunarodnoj World Wide Web konferenciji. Iste godine osnuju i tvrtku Google. Na PageRanku i dan danas počivaju rezultati Google pretrage.

2 Definicije

Modeliramo strukturu mreže od n web stranica kao usmjereni graf gdje vrhovi predstavljaju web stranice, a brid iz vrha u u vrh v predstavlja poveznicu (link) na stranici u koja pokazuje na stranicu v. Graf opisujemo matricom G, koja ima 1 na mjestu (i, j) ako vrh i ima link na vrh j. No, željeli bismo vrhove rangirati po tome koliko su važni vrhovi koji na njih pokazuju, uz uvjet da na što više drugih vrhova određeni vrh, važnost istog se proporcionalno smanjuje. Ako s x_i označimo važnost vrha i, te s n_i broj vrhova na koje pokazuje vrh i te L_j skup vrhova koji pokazuju na vrh j, zapravo želimo da vrijedi

$$x_j = \sum_{i \in L_j} \frac{1}{n_i} x_i$$

Zatim definiramo matricu A koja to uzima u obzir: za $[A_{ij}]$ stavimo $1/n_i$ ako vrh i pokazuje na vrh j, a 0 inače. Tako vidimo da je problem zapravo naći vektor x t.d. $Ax = x$, tj. naći svojstveni vektor matrice A kojem pripada svojstvena vrijednost 1.

3 Markovljevi lanci i PageRank

Neka je skup svih stanja u kojima se neki proces može nalaziti konačan s n elemenata te svako stanje označimo brojem od 1 do n. Ako buduće stanje procesa ovisi samo o sadašnjem stanju, a ne i o prošlima (npr. koju ćemo sljedeću internetsku stranicu posjetiti ovisi samo o stranici koju trenutno gledamo, ali ne i o stranicama koje smo nekoć gledali), tada se takav proces naziva Markovljev lanac. Ako se u sadašnjosti nalazimo u stanju i, označimo s a_{ij} vjerojatnost da ćemo se u sljedećem trenutku nalaziti u stanju j. Označimo s A matricu $[a_{ij}]$. Iz stanja i ćemo u sljedećem trenutku gotovo sigurno prijeći u neko stanje iz $\{1, \dots, n\}$ pa je zbroj svakog retka u matrici A jednak 1, a svaki $[a_{ij}]$ je nenegativan (jer predstavlja vjerojatnost prijelaza iz stanja i u stanje j). Takve matrice zovu se stohastičkima.

Svaki proces ima neku svoju početnu distribuciju, a u ovom slučaju ona se reprezentira vektorom $\lambda = (\lambda_1, \dots, \lambda_n)$ gdje λ_i predstavlja vjerojatnost da se nalazimo u stanju i. Stoga je to vektor čiji su svi elementi nenegativni i u zbroju daju 1 (jer ćemo se gotovo sigurno nalaziti u nekom stanju). Pretpostavimo da su na početku sva stanja jednaka, tj. $\lambda_1 = \dots = \lambda_n = 1/n$, tj. s jednakom vjerojatnošću gledamo svaku stranicu na našem internetu. U sljedećem trenutku vjerojatnost da gledamo stranicu j je $\lambda_1 a_{1j} + \lambda_2 a_{2j} + \dots + \lambda_n a_{nj}$, odnosno nova distribucija je upravo λA. Distribuciju za koju vrijedi $\lambda = \lambda A$ nazivamo stacionarna distribucija jer u svakom koraku vjerojatnost da se nalazimo u nekom stanju ostaje ista. Želimo naći distribuciju u beskonačnosti, tj. takav λ za kojeg je λ_i limes po n vjerojatnosti da se u n-tom trenutku nalazimo u stanju i. Lako se pokaže da ako takva distribucija postoji, ona nužno mora biti stacionarna.

Označimo s $L(i)$ skup svih stranica na koje postoji link iz stranice i. Promatramo "slučajnog surfera" koji iz stranice i s nekom vjerojatnošću $\alpha \in$ (0,1) prijeđe u slučajno odabranu stranicu $j \in L(i)$, a s vjerojatnošću 1 - α u slučajno odabranu stranicu (bilo koju). Neka je π_i asimptotski postotak vremena koje će surfer provesti na stanici i. Originalni PageRank određuje važnost stranice i kao π_i.

4 Viseći vrhovi

Visećim vrhovima (eng. *dangling nodes*) zovemo one vrhove i koji nemaju nijedan izlazni brid, tj. vrijedi $n_i = 0$. Tada je i-ti redak u A nul redak. Budući da daljnji algoritam počiva na tome da imamo stohastičku matricu, glavno je pitanje što napraviti s tim nul retcima odn. visećim vrhovima. Jedno moguće rješenje je dodati izmišljene veze s visećih vrhova, tj. dopuniti nul retke na način da definiramo matricu $M_\alpha = (1-\alpha)A + \alpha S$ za neki $\alpha \in [0,1]$, gdje je $[S_{ij}] = 1/n$. Tada je M uistinu stohastička i to je algoritam obrađen u *prez. s predavanja* s priloženim kodom - svojstvenom vektoru konvergiramo primjenom matrice na neki stohastički vektor.

5 Sažimanje (*lumping*)

Kako je u praksi udio visećih vrhova izuzetno velik (po Ipsen and Selee (2007) on može doseći čak i 80% ukupnog broja vrhova), prirodno je zapitati se što s njima možemo efikasno učiniti. Grupirat ćemo matricu na način da imamo puno manju podmatricu samo s nevisećim vrhovima. Preuredimo prvo polaznu matricu tako da sortiramo vrhove. Neka je, od n vrhova, k broj nevisećih ($1 \le k < n$). Tih k vrhova fiksirajmo kao prvih k. Sada polazna $n \times n$ matrica izgleda ovako:

$$H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}, \tag{1}$$

gdje je H_{11} $k \times k$ matrica linkova između nevisećih vrhova a H_{12} sadrži linkove nevisećih na viseće vrhove. Time smo izdvojili viseće vrhove u donjih $n-k$ redova, a gornjih k redova su i dalje stohastički.

Ideja se sada svodi na sljedeće: o rezultatu (poretku) nevisećih k vrhova informaciju imamo iz matrice H_{11}, a na poredak $n - k$ visećih vrhova zapravo samo utječu oni vrhovi koji pokazuju na njih, a ta informacija je u H_{12}.

Postupak će početi naizgled isto, tj. prvo ćemo odabrati stohastički vektor w (2) s kojim ćemo zamijeniti nul retke, a zatim napraviti i konveksnu kombinaciju sa stohastičkim vektorom prilagodbe v (3) kako bismo osigurali jedinstvenost stacionarne distribucije.

$$S := \begin{bmatrix} H_{11} & H_{12} \\ ew_1^T & ew_2^T \end{bmatrix}, \text{ gdje je } w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \text{ dimenzije } n \times 1 \tag{2}$$

$$G := \alpha S + (1-\alpha)ev^T,\ 0 \leq \alpha < 1) \tag{3}$$

Konačno, matrica G izgleda ovako:

$$G := \begin{bmatrix} G_{11} & G_{12} \\ eu_1^T & eu_2^T \end{bmatrix}, \tag{4}$$

gdje je $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ konveksna kombinacija v i w.
Za takvu matricu G rezultati iz Ipsen and Selee (2007) osiguravaju da ćemo moći sažeti viseće vrhove u jedan vrh i pritom očuvati točnost rješenja.

6 Analiza složenosti

7 Tehnički detalji algoritma i empirijski rezultati

Program implementiramo u programskom jeziku *Python* koristeći module *numpy* i *scipy* za potrebne metode za efikasno baratanje velikim matricama. Izvorni kod priložen je radu u datoteci *PageRankWDanglingNodes.py*. U mapi podaci dano je više skupova podataka za testiranje preuzetih iz Sahu, a kod je testiran na primjeru *enron*.

Kod je komentiran i čitljiv, pa navodimo samo nekoliko tehničkih detalja:

- Sortiranje vrhova u matrici H radimo na način da pamtimo permutaciju vrhova iz originalnog poretka (zato nam koristi i funkcija za inverz permutacije).

- Za matricu A (s originalnim podacima) i zatim H, budući da su rijetko popunjene matrice, koristimo *sparse* paket iz *scipy* modula. Stvaramo ih kao objekt klase `coo_matrix` (*COOrdinate format*) budući da im dajemo poznate elemente s koordinatama, a zatim ih pretvaramo u `csc_matrix` format (*Compressed Sparse Column format*) za učinkovitije izvođenje operacija.

- Implementiran je algoritam sa sažimanjem, a nakon njega i algoritam bez sažimanja koji djeluje na cijelu matricu H. Za oba algoritma izmjereno je i ispisano vrijeme izvođenja.

Empirijski vidimo da se kod sa sažimanjem izvodi u prosjeku 3 do 4 puta brže od koda bez sažimanja. To je potvrda slutnje analize složenosti.

References

Ilse Ipsen and Teresa Selee. PageRank Computation, with Special Attention to Dangling Nodes. *SIAM Journal*, 2007. URL `https://www.researchgate.net/publication/220656288_PageRank_Computation_with_Special_Attention_to_Dangling_Nodes`.

Subhajit Sahu. LAW Graphs Part 1 (A-U). URL `https://www.kaggle.com/datasets/wolfram77/graphs-law-01/`.