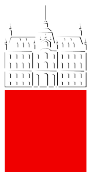
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Katzova središčnost in Googlov PageRank

Projekt pri predmetu Finančni praktikum

Anamari Oštarijaš, Tina Ražić

Ljubljana, december 2018

Kazalo

1 Opis projekta

Kompleksna omrežja lahko analiziramo z uporabo različnih kvantitetnih merjenj, imenujemo jih tudi mere središčnosti, ki intuitivno zajamejo pomembnost določenih vozlišč. V projektu bova implementirali Googlov PageRank in Katzovo središčnost z uporabo potenčne metode. Na različnih grafih (tudi socialnih omrežjih) bova analizirali in primerjali, kako merjenji razvrstita vozlišča po pomembnosti.

2 Katzova središčnost

2.1 Uvod

Katzova središčnost je ena izmed številnih mer središčnosti na grafih. Predstavil jo je Leo Katz leta 1953. Uporabljamo jo pri analiziranju socialnih omrežji, saj lahko z njo izmerimo relativno stopnjo vpliva posameznega igralca, ki predstavlja eno vozlišče v grafu. Tipično mere središčnosti pri merjenju upoštevajo le najkrajšo pot med dvema igralcema, Katzova središčnost pa meri vpliv glede na celotno število sprehodov med dvema igralcema.

2.2 Opis Katzove središčnosti

Katzova središčnost izmeri vpliv igralca v omrežju, tako da upošteva direktne sosede igralca in vse druge igralce, ki so posredno povezani s tem igralcem, preko njegovih direktnih sosedov. Naj bo naše omrežje graf z n vozlišči oziroma igralci. Vsaka povezava v grafu dobi utež α in z α^d izračunamo težo povezave vozlišča z durgim vozliščem, pri čemer je d število povezav, ki ju povezuje. Naš graf predstavimo z matriko sosednosti A, torej element matrike a_{ij} ima vrednost 1, če je vozlišče i povezano z vozliščem j in 0, če nista povezana. Potence matrike A nam povejo, če je vozlišče povezano s drugimi indirektnimi vozlišči preko sosedov. Na primer: če je v matriki A^3 element $a_{2,5} = 1$,pomeni, da sta vozlišče 2 in vozlišče 5 povezana s tremi povezavami preko sosedov prve stopnje in sosedov druge stopnje. Označimo s $C_{Katz}(i)$ Katzovo središčnost vozlišča i. Potem lahko izračunamo središčnost na sledeči način:

$$C_{Katz}(i) = \sum_{k=1}^{\infty} \sum_{j=1}^{n} \alpha^k (A^k)_{ij}$$

Pri izbiri α moramo upoštevati zgornjo omejitev

$$\alpha < \frac{1}{|\lambda_{max}|}.$$

3 Googlov PageRank

Nemogoče je definirati splošno mer pomembnosti, ki bi bila sprejemljiva za vse uporabnike iskalnika. Google uporablja pagerank za mero kakovosti spletnih strani. Temelji na predpostavki, da število linkov (povezav) do in iz strani daje informacijo o pomembnosti strani.

Naj bodo vse spletne strani urejene s števili od 1 do n in naj bo i neka spletna stran. Potem O_i določa množico strani, s katerimi je i povezana, tako da i vsebuje link do strani v množici O_i (*outlink*). Število outlinkov označimo z $N_i = \|O_i\|$. Množica *inlinkov*, označena z I_i, je množica strani, ki imajo outlink do i (strani v I_i vsebujejo linke do i). Splošno, več ko ima stran$i$ inlinkov, pomembnejša je.
Vseeno pa bi bilo s takim sistemom preprosto manipulirati (če bi nekdo želel, da njegovo spletno stran vidi čim več ljudi, bi ustvaril veliko število nepomembnih spletnih strani, ki bi vsebovale linke do njegove spletne strani). Da bi tako manipulacijo preprečili, definiramo rang vozlišča i tako, da če ima visoko rangirana stran j outlink do i, to doda pomembnosti i na sledeč način: rang strani i je utežena vsota rangov strani, ki imajo outlink do i. Obteženost je taka, da je rang strani j razdeljen enakomerno med njenimi outlinki. Z enačbo:

$$r_i = \sum_{j \in I_1} \frac{r_j}{N_j}.$$

Ta definicija je rekurzivna, zato pageranki ne morejo biti izračunani direktno. Uporabimo iteracijo. Najprej ugibamo začetni rangni vektor r^0. Potem iteriramo:

$$r_i^{(k+1)} = \sum_{j \in I_1} \frac{r_j^{(k)}}{N_j}$$

Raje zapišimo problem z matrikami. Naj bo Q kvadratna matrika dimenzije n. Definiramo:

$$Q_{ij} = \begin{cases} 1/N_j & \text{, če obstaja link od j do i} \\ 0 & \text{, sicer} \end{cases}$$

Torej ima vrstica i neničelne elemente na mestih inlinkov i. Podobno ima stolpec j neničelne elemente enake N_j na mestih outlinkov j, vsota vseh elementov v stolpcu je enaka 1. Definicija je ekvivalentna skalarnem produktu vrstice i in vektorja r, ki vsebuje range vseh strani.
Enačbo sedaj lahko zapišemo v matrični obliki:

$$\lambda r = Qr, \qquad \lambda = 1,$$

Tako dobimo, da je r lastni vektor matrike Q z lastno vrednostjo $\lambda = 1$. Sedaj preprosto vidimo, da je iteracija ekvivalentna

$$r^{(k+1)} = Qr^{(k)}, \qquad k = 0, 1, \ldots,$$

kar je potenčna metoda za izračun lastnega vektorja.

Problem: Ni jasno, da je pagerank dobro definiran, saj ne vemo ali obstaja lastna vrednost enaka 1. Pomagamo si s teorijo Markovske verige.

Obstaja interpretacija pageranka z naključnimi sprehodi. Predpostavimo, da uporabnik na spletni strani izbere naslednjo stran med outlinki z enako verjetnostjo. Markovska veriga je slučajni proces v katerem je naslednje stanje določeno le s trenutnim stanjem, proces nima spomina. Prehodna matrika Markovske verige je Q^T.

Slučajni uporabnik nikoli ne sme obtičati. Z drugimi besedami, naš model slučajnega sprehoda ne sme imeti strani brez outlinkov (taka stran bi imela stolpec iz ničel v Q). Torej je model prilagojen tako, da so ničelni stolpci nadomeščeni s konstantnimi vrednostmi na vseh mestih. To pomeni, da je enaka verjetnost, da pridemo na katero koli drugo spletno stran.

Definiramo vektorje:

$$d_j = \begin{cases} 1 & ,\text{če} N_j = 0 \\ 0 & ,\text{sicer} \end{cases}$$

za $j = 1, .., n$ in

$$e = [1 \dots 1]^T \in \mathbb{R}^n.$$

Prilagojena matrika je definirana s $P = Q + \frac{1}{n} e d^T$. S to prilagoditvijo je matrika P desna stohastična matrika; vsi elementi so nenegativni in elementi vsakega stolpca se seštejejo v 1.

Desna stohastična matrika P zadošča

$$e^T P = e^T$$

.
Želimo definirati pagerank vektor kot enoličen lastni vektor matrike P z lastno vrednostjo 1:

$$Pr = r$$

.
Lastni vektor prehodne matrike je stacionarna verjetnostna porazdelitev Markovske verige. Element na mestu i (r_i) je verjetnost, da po velikem številu korakov slučajni uporabnik pristane na strani i. Za zagotovitev enoličnosti mora biti matrika ireducibilna (slučajni uporabnik se lahko v nekem delu grafa zagozdi, v tem primeru ima matrika več lastnih vrednosti enakih 1).
Edinstvenost največje lastne vrednosti ireducibilne, pozitivne, desne stohastične matrike je zagotovljena s *Perron-Frobeniusovim izrekom*, največja singularna vrednost bo enaka 1, pripadajoč slastni vektor je pozitiven in je edini lastni vektor, ki je nenegativen.

Zaradi velikosti spleta je matrika linkov P reducibilna, torej pagerank lastni vektor ni dobro definiran. Da si zagotovimo ireducibilnost, umetno dodamo linke iz vsake spletne strani do vseh drugih. To lahko storimo, če vzamemo konveksno kombinacijo P in matrike ranga 1:

$$A = \alpha P + (1 - \alpha) \frac{1}{n} e e^T,$$

za nek α, ki zadošča $0 \leq \alpha \leq 1$. Matrika A je desna stohastična. Razlaga naključnega sprehoda dodatnega rang-1 izraza je, da bo uporabnik na vsakem časovnem koraku skočil na naključno stran z verjetnostjo $1 - \alpha$. Za konvergenco numeričnega algoritma za lastno vrednost je pomembno, da vemo, kako so s prilagoditvijo ranga-1 spremenjene lastne vrednosti.

Izrek: *Naj bodo lastne vrednosti desne stohastične matrike P enake* $1, \lambda_2, \lambda_3, \dots, \lambda_n$.

Potem so lastne vrednosti $A = \alpha P + (1-\alpha)\frac{1}{n}ee^T$ *enake* $1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n$.

Ta izrek nam pove, da tudi če ima P več lastnih vrednosti enakih 1, kar je po navadi res, bo druga največja lastna vrednost matrike A enaka α.
Namesto prilagoditve lahko definiramo:

$$A = \alpha P + (1-\alpha)ve^T,$$

Kjer je v nenegativen vektor z normo 1, ki je lahko izbran tako, da je iskanje pristransko do strani določene vrste. Zato ga včasih imenujemo *personaliziran vektor*. Uporaben je tudi v izogib manipulacijam proti t.i.*link farms*.

3.1 Potenčna metoda

Želimo rešiti problem lastne vrednosti:

$$Ar = r,$$

Kjer je r nomaliziran $\|r\|_1 = 1$. Iskani lastni vektor označimo s t_1.
Prepostavimo, da imamo dan začetni približek $r(0)$.

Algoritem:

$$\text{za k} = 1, 2, \ldots \text{ do konvergence}$$

$$q^{(k)} = Ar(k-1)$$

$$r^{(k)} = q^{(k)}/\|q^{(k)}\|_1$$

Konvergenca je odvisna od porazdelitve lastnih vrednosti. Če je druga največja lastna vrednost blizu 1, bo iteracija zelo počasna. To na srečo ne velja za Google matriko. Vektor normaliziramo, da ne bi z iteracijami postali preveliki ali premajhni, posledično nereprezentativni s števili s plavajočo vejico. To v resnici ni potrebno, saj se v primeru desnih stohastičnih matrik temu izognemo. En izračun pageranka lahko traja več dni.

4 Načrt dela

Najprej bova oba algoritma preizkusili na manjših grafih za zagotovitev pravilnosti, kasneje pa še na omrežjih. Primerjali bova razlike v rangiranju in čase izračunov na večjih in manjših grafih, da vidiva, kako velikost omrežja vpliva na vsako metodo.
Algoritma bova napisali v programu Python s pomočjo knjižnice NetworkX za generiranje omrežij.