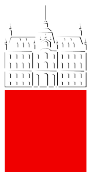
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Katzova središčnost in Googlov PageRank

Kratko poročilo, Projekt pri predmetu Finančni praktikum

Anamari Oštarijaš, Tina Ražić

Ljubljana, december 2018

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

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

Ta metoda 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.
Da bi preprečili možne manipulacije, 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.
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.

Predpostavili smo, da obstaja lastna vrednost enaka 1. Ta predpostavka je pojasnjena s teorijo slučajnih sprehodov, podrobnosti so napisane v daljši dokumentaciji (*katzgooglepagerankdefinition*).

3.1 Prilagoditev modela

Da naša matrika zadošča pogojem *Perron-Frobeniusovega izreka*, torej, da je le ena lastna vrednost enaka 1, jo moramo ustrezno prilagoditi. Matrika mora biti ireducibilna, pozitivna in desna stohastična.
Ničelne stolpce v matriki Q nadomestimo s konstantnimi vrednostmi na vseh mestih (s tem rešimo problem strani, ki ne vsebujejo nobenih outlinkov).
Definiramo vektorje:

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

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

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

Prilagojena matrika je definirana s $P = Q + \frac{1}{n} e d^T$,desno stohastično matrika, ki zadošča:

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

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

$$Pr = r$$

.
Element r_i je verjetnost, da po velikem številu korakov slučajni uporabnik pristane na strani i.
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:

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

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 $1 - \alpha$. Če so bile lastne vrednosti desne stohastične matrike P enake $1, \lambda_2, \lambda_3, ..., \lambda_n$, bodo lastne vrednosti matrike A enake $1, \alpha\lambda_2, \alpha\lambda_3, ..., \alpha\lambda_n$. Tako si zagotovimo, da je le ena lastna vrednost enaka 1.

3.2 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.