Social and Graph Data Management
Node and Link Analysis

December 10th, 2021

M2 Data Science

Estimating Node Worth

Nodes on the Web: pages (sites, Wikipedia, ...), users (Twitter, Facebook), etc.

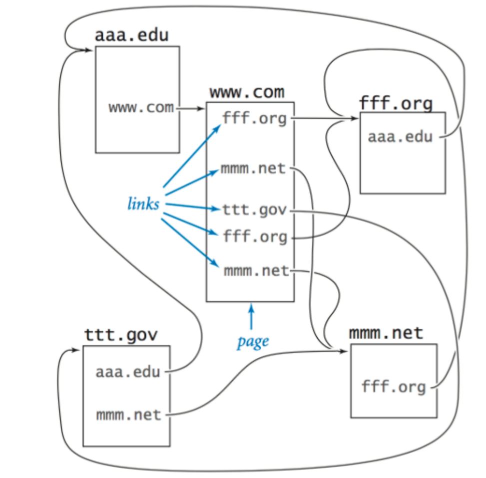
To use/find the nodes that are more “interesting” than others, we have to estimate the worth of each node.

Can be used combined with textual (or profile) information to retrieve content in information retrieval – but also the *links* between information are important

PageRank: Ranking Nodes in A Graph

Web:

- be a fixed set of pages each page containing a fixed set of hyperlinks and each link a reference to some other page
- Intuition: A page with multiple paths to it is a important page.



Random Surfer Model

- The goal to be achieved for web search engines: is to rank web pages containing keywords according to the importance of the page.
- Importance/Rank (*PageRank*) of a web page p_i: the probability that a random surfer will arrive at page i.
    - A *web surfer* can *move* randomly from one site to another by typing the site name or simply clicking a link on the page being viewed.

Random Surfer Model

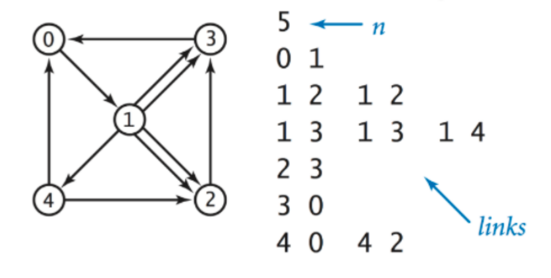
90–10 rule captures both methods of moving to a new page:

- 90 percent of the time the random surfer clicks a random link on the current page (each link chosen with equal probability)
- 10 percent of the time the random surfer goes directly to a random page (all pages on the web chosen with equal probability).

Given a set of n web pages numbered 0 through $n-1$ along with information about the hyperlinks contained in each page. Calculates the probability that a web surfer starting at page 0 will arrive at page i after m times.

Input format

- We assume that there are n web pages, numbered from 0 to $n-1$, and we represent links with ordered pairs of such numbers, the first specifying the page containing the link and the second specifying the page to which it refers.
- The input format we adopt is an integer (the value of n) followed by a sequence of pairs of integers (the representations of all the links).



Transition matrix

We use a two-dimensional matrix, which we refer to as the transition matrix, to completely specify the behavior of the random surfer. With n web pages, we define an n-by-n matrix such that the entry in row ***i*** and column ***j*** is the probability that the random surfer moves to page ***j*** when on page ***i***.

$$\overset{\textit{leap probabilities}}{\begin{bmatrix} .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \\ .02 & .02 & .02 & .02 & .02 \end{bmatrix}} + \overset{\textit{link probabilities}}{\begin{bmatrix} 0 & .90 & 0 & 0 & 0 \\ 0 & 0 & .36 & .36 & .18 \\ 0 & 0 & 0 & .90 & 0 \\ .90 & 0 & 0 & 0 & 0 \\ .45 & 0 & .45 & 0 & 0 \end{bmatrix}}$$

$$= \overset{\textit{transition matrix}}{\begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & 92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix}}$$

Two moves

Calculate the probability that a user moves from page k to page j after two moves.

- Calculate the product of the probability of moving from k to i with the probability of traversing i to j for all possible values of i, and add the results.
- This probability matrix $k, j = \overline{1, n}$ is the result of the matrix multiplication $p[][] \times p[][]$.

p

$$
\begin{matrix}
.02 & .92 & .02 & .02 & .02 \\
.02 & .02 & .38 & .38 & .20 \\
.02 & .02 & .02 & .92 & .02 \\
.92 & .02 & .02 & .02 & .02 \\
.47 & .02 & .47 & .02 & .02
\end{matrix}
$$

probability of surfing from i to 2 in one move

probability of surfing from 1 to i in one move

p^2

$$
\begin{matrix}
.05 & .04 & .36 & .37 & .19 \\
.45 & .04 & \boxed{.12} & .37 & .02 \\
.86 & .04 & .04 & .05 & .02 \\
.05 & .85 & .04 & .05 & .02 \\
.05 & .44 & .04 & .45 & .02
\end{matrix}
$$

probability of surfing from 1 to 2 in two moves (dot product)

m moves

- Calculate the probability after 3 moves can be done by multiplying the resulting matrix p^2 again by the matrix $p[][]$ and so on.
- However, the calculation of multiplying by 2 matrices is quite *'expensive'* especially when n is usually very large.

first move

$$
\begin{array}{ccccc}
\texttt{ranks[]} & & \texttt{p[][]} & & \texttt{newRanks[]} \\
[\ 1.0\ \ 0.0\ \ 0.0\ \ 0.0\ \ 0.0\ ] & * & \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix} & = & [\ .02\ \ .92\ \ .02\ \ .02\ \ .02\ ]
\end{array}
$$

probabilities of surfing from 0 to i in one move

m moves

second move

probabilities of surfing from 0 to i in one move

probabilities of surfing from i to 2 in one move

probability of surfing from 0 to 2 in two moves (dot product)

$$[\ \boxed{.02\ \ .92\ \ .02\ \ .02\ \ .02}\ ] * \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix} = [\ .05\ \ .04\ \ \boxed{.36}\ \ .37\ \ .19\ ]$$

probabilities of surfing from 0 to i in two moves

third move

probabilities of surfing from 0 to i in two moves

$$[\ .05\ \ .04\ \ .36\ \ .37\ \ .19\ ] * \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix} = [\ .44\ \ .06\ \ .12\ \ .36\ \ .03\ ]$$

probabilities of surfing from 0 to i in three moves

m moves

20th move

probabilities of surfing from 0 to i in 19 moves

$$
[\ .27\ \ .26\ \ .15\ \ .25\ \ .07\ ] * \begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & .92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix} = [\ .27\ \ .26\ \ .15\ \ .25\ \ .07\ ]
$$

probabilities of surfing from 0 to i in 20 moves (steady state)

Variants of PageRank

Depending on where the surfer teleports with probability $1 - \alpha$, we have different variants of PageRank:

- classic PageRank: the surfer can jump to any node.
- personalized PageRank: the surfer can only jump to their start page
- topic-sensitive PageRank: the surfer can only jump to a set of same-topic pages

Table of contents

The Link Prediction Problem

Social networks are evolving, and new relationships (links) appear all the time

Link Prediction Problem: predict which links are more likely to appear in a social network

Assumes that links can be predicted via analysis based only on the social network itself

Applications:

- new link recommendation (e.g., new friends)
- missing link inference
- analyzing network evolution

Link Scoring Function

We want to "guess" the score of potential links for a graph $G = (V, E)$, i.e., a function defined on the missing links $E' = (V \times V) \backslash E$:

$$\text{score} : E' \rightarrow \mathbb{R}^+$$

For a given i, score(i, j) established a ranking of all (unliked) nodes j relative to i – best scores are the most likely new links

How can we define the score function using only the properties intrinsic to the network?

Node Neighbourhood Scores

- Common Neighbours, most straightforward counts the number of common neighbors:

$$\text{score}(i, j) = |N(i) \cap N(j)|$$

- Jaccard coefficient, computes the "similarity" between the neighborhood sets

$$\text{score}(i, j) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$$

- Preferential attachment, the score is proportional to the degrees of each node:

$$\text{score}(i, j) = k_i k_j$$

Path-Based Scores

- Inverse Distance, the score is inversely proportional to the distance between two nodes

$$\mathrm{score}(i,j) = 1/d_{ij}$$

- Katz, where the score is a weighted sum of all the paths between i and j

$$\mathrm{score}(i,j) = \sum_{l=1}^{\infty} \beta^l |\mathrm{paths}_{i,j}^{\langle l \rangle}|,$$

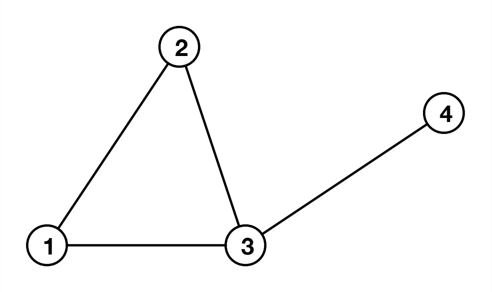
where $\beta \in (0,1)$

Random Walk-Based Scores

- Personalized PageRank, generally any PageRank-related measure in which the teleportation vector is rooted at i
- SimRank, a recursive definition based on the score of neighbors

$$\text{score}(i, j) = \gamma \frac{\sum_{a \in N(i)} \sum_{b \in N(j)} \text{score}(a, b)}{k_i k_j}$$

Applying the Scores – Example



Acknowledgments

Figures in slides 6 to 11 are taken from the course "Introduction to Programming", Princeton University

References i