Projeto 5
Como o Google Ordena Páginas

1 Introdução

Quando fazemos uma procura em um site de busca na internet, como o Google, desejamos obter as páginas que contém um determinado assunto ou palavra-chave, ordenadas em ordem decrescente de prioridade, apesar de não haver uma definição muito clara do que isso significa. O aparecimento do Google no final dos anos 90 foi uma espécie de divisor de águas no que se refere a procura de assuntos na rede. Isto porque, o Google parece sempre colocar os sites mais relevantes primeiro. Com outros sites de procura, muitas vezes era necessário olhar páginas e mais páginas até que os resultados interessantes aparecessem.

O objetivo deste projeto é entender como funciona um site de procura. Em particular descreveremos o algoritmo utilizado pelo Google para ordenar as páginas em ordem decrescente de importância. Um site de procura como o Google basicamente faz 3 coisas:

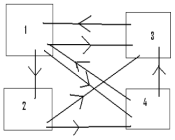
i) varre a rede e localiza todas as páginas públicas;

ii) indexa os dados de i) em um banco de dados de forma que uma procura por palavra-chave possa ser feita de uma maneira eficiente;

iii) atribui uma importância a cada página do banco de dados de ii), de forma que quando um usuário faz uma procura e o subconjunto das páginas que contém um determinado termo é encontrado, elas podem ser listadas em ordem decrescente de importância;

Como foi dito, descreveremos como iii) acima é feita e implementaremos um algoritmo para realizar tal atribuição de importâncias. A idéia básica é atribuir pesos positivos às diversas páginas, sendo as com maior peso, as mais importantes. Para isso, é preciso armazenar a rede como um grafo orientado, onde os vértices são as diversas páginas e as arestas orientadas (setas) representam um link de uma página para outra.

Olhemos para a seguinte rede de 4 páginas. A página 1 possui links para as páginas 2,3 e 4. A página 2 para 3 e 4, a página 3 para 1 e a página 4 para 1



e 3. A importância da página i, onde $i \in \{1, 2, 3, 4\}$, será representada por um número real positivo x_i, o peso da página i. A questão a se pensar agora é, como atribuir os x_i's? Existem alguns fatores a se levar em conta, mais precisamente:

1. o fato de uma página ter muitos links apontando para ela a torna mais importante;

2. apesar do que foi dito acima, os links que apontam para uma certa página fixada P, em geral têm importâncias diferentes: quando uma página que possui muitos links apontando para ela aponta para P, isto deve pesar mais na importância de P que links vindos de páginas pouco apontadas;

3. por fim, um link vindo para P de uma página que aponta para muitas outras páginas deve ter menor importância que um link vindo de páginas que apontam para poucas.

A motivação para os fatores acima vem do seguinte: uma página P será importante, se um usuário clicando aleatoriamente nos links das páginas que ele encontra, tiver alta probabilidade de acessar P.

Vamos agora, através do exemplo, descrever como atribuir os pesos de acordo com os fatores qualitativos acima descritos:

$$\begin{cases} x_1 = x_3/1 + x_4/2 \\ x_2 = x_1/3 \\ x_3 = x_1/3 + x_2/2 + x_4/2 \\ x_4 = x_1/3 + x_2/2 \end{cases}$$

Observe que definimos o peso de uma página como a soma dos pesos das páginas que apontam para ela, divididos pelo número de links que saem de cada página. Obtemos um sistema linear que, em forma matricial, é dado por:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \tag{1}$$

ou seja transformamos o problema de ranqueamento da importância das páginas na rede num problema de encontrar uma solução para (1).

Note que a equação acima pode ser escrita na forma $x = Ax$. A matriz A é denominada matriz de ligação. Como a soma dos elementos de cada coluna de A é igual a 1, pode-se provar que (1) tem solução. Resolvendo-se o problema obtemos as soluções, que são todas mútiplos do vetor normalizado (soma das componentes igual a 1)

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.387 \\ 0.129 \\ 0.290 \\ 0.194 \end{pmatrix}.$$

Note que a página 3 apesar de muito apontada, não é a mais importante. Isto porque ela aponta somente para a 1, que ainda é apontada pela 4, o que a torna a mais vista por um clicador de links aleatório.

Por construção, a soma de cada coluna de uma matriz de ligação é igual a 1[1], garantindo que o problema $Ax = x$ sempre tenha solução. Outras propriedades desejáveis são a existência de uma solução cujas componentes são positivas tal que qualquer outra solução do problema seja um múltiplo dela. Com isso, normalizando-se esta solução, garantimos a existência de únicos pesos (normalizados) x_i para cada página.

Veremos adiante que uma perturbação do problema sempre pode ser feita de maneira a garantir essas propriedades, nos levando a formular um problema

$$Mx = x, \tag{2}$$

onde M é a matriz do problema perturbado.

2 Alguns resultados matemáticos

Enunciaremos nesta seção, sem demonstrar, alguns resultados de álgebra linear.

Lema 1. Se M é uma matriz $n \times n$ tal que a soma dos elementos de cada coluna é igual a 1, então 1 **é autovalor** de M, o que é o mesmo que afirmar que a equação (2) tem solução. Uma tal solução é chamada de **autovetor** associado ao autovalor 1.

Teorema 1. (**Perron-Frobenius**)
Seja M uma matriz $n \times n$ com todos os elementos positivos ($M_{ij} > 0$, $\forall i, j \in \{1, \dots n\}$) e colunas com soma 1 ($\sum_{i=1}^{n} M_{ij} = 1$, $\forall j$). Então 1 é o autovalor de maior módulo de M, seu auto-espaço é unidimensional[2] e o autovetor normalizado só tem entradas estritamente positivas.

O fato de 1 ser o autovalor de maior módulo garante a convergência do algoritmo que será apresentado na próxima seção. Uma matriz de ligação A, em geral, não satisfaz as hipóteses do teorema de Perron-Frobenius, pois pode ter elementos nulos. Substituímos então A pela matriz perturbada

$$M = (1 - m)A + mS, \tag{3}$$

onde $0 < m < 1$ e S é a matriz $n \times n$ com todas as entradas iguais a $1/n$.

Esta é (ou era) a estratégia usada pelo Google, que adota $m = 0.15$ e **esse será o valor usado no projeto**. Essa correção feita na matriz A torna todas as entradas de M estritamente positivas e a soma dos elementos em cada coluna continua sendo igual a 1. Uma observação importante é que isso ocorre para todo $0 < m < 1$. Assim, uma perturbação pequena de A tem todas as entradas maiores que zero, com a mesma propriedade de soma 1 ao longo das colunas, satisfazendo as hipóteses do teorema de Perron-Frobenius.

Assim, dada uma rede sem páginas mortas, o processo de ranquear as páginas consiste em obter a matriz A, depois calcular M e a partir de M, achar x. O que veremos a seguir é uma forma muito rápida (computacionalmente) de achar a distribuição de pesos x.

[1] Desde que a rede não possua páginas que não apontem para lugar nenhum, o que assumiremos na nossa formulação.

[2] Todos os autovetores são múltiplos de um mesmo vetor.

3 Cálculo do vetor x

Dado um vetor v com n componentes, defina a sua norma por

$$\|v\| = \sum_{i=1}^{n} |v_i|.$$

O algoritmo para o cálculo do vetor de pesos é descrito como:

- Escolha um vetor $x^{(0)}$ **normalizado** ($\|x^{(0)}\| = 1$) e todo positivo ($x_i^{(0)} > 0$, $1 \leq i \leq n$). Por exemplo, o vetor com todas as componentes iguais a $1/n$.
- Para $k = 1, 2, 3, \ldots$ calcule

$$x^{(k)} = Mx^{(k-1)}$$

Este é o famoso Método das Potências[3]. Se a matriz M satisfaz as hipóteses to Teorema 1, pode-se mostrar que $x^{(k)}$ converge para o autovetor normalizado associado ao autovalor 1 com a taxa

$$c = max_{1 \leq j \leq n} \left| 1 - 2 \min_{1 \leq i \leq n} M_{ij} \right|$$

e temos ainda a estimativa de erro *a posteriori*

$$\|x - x^{(k)}\| \leq \frac{c}{1-c} \|x^{(k)} - x^{(k-1)}\|, \tag{4}$$

onde x é o autovetor normalizado.

4 Exercícios

1. Seja M uma matriz $n \times n$ satisfazendo as hipóteses do Teorema 1. Mostre que se y é um vetor com todas as entradas positivas e normalizado, então $z = My$ será um vetor com todas as entradas positivas e normalizado.

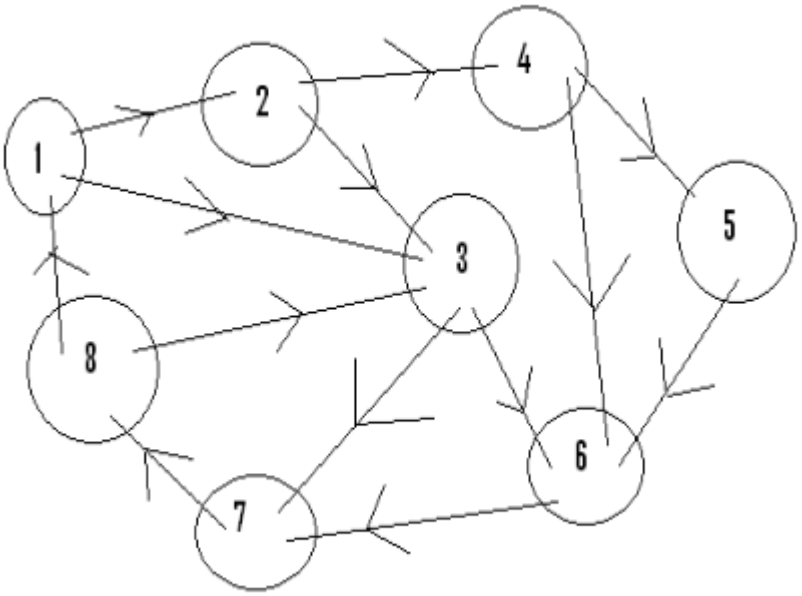
2. Considere agora uma matriz M da forma (3). Se y é o vetor com n entradas iguais a $1/n$, mostre que

$$My = (1-m)Ay + m \begin{pmatrix} 1/n \\ 1/n \\ \vdots \\ 1/n \end{pmatrix}.$$

Mostre também que se y for um vetor cujas entradas somam 1, então My também é calculado pela expressão acima.

3. Escreva um programa que, dada uma matriz de ligação A $n \times n$, calcula o vetor dos pesos normalizado conforme o algoritmo descrito na Seção 3 e ordena as "páginas"de acordo com ele. Use $m = 0.15$ e a condição inicial $x^{(0)} = (1/n, 1/n, \ldots, 1/n)^t$. Calcule as iterações até que o erro dado por (4) seja menor do que 10^{-5} (para isso, você deve calcular a constante c, que deve ser impressa também). Teste o seu programa com o exemplo relacionado à figura da próxima página.

[3] Veja o livro de Anton & Rorres.



Representação esquemática para o teste do Exercício 3

Referências

1. Howard Anton, Chris Rorres, *Álgebra Linear (com aplicações)*, Bookman, Porto Alegre, 2012

2. K. Brian, T. Laise, *The $25,000,000,000 Eigenvector: The Linear Algebra behind Google5*, SIAM Review, vol. 48, No. 3, pp. 569–581, 2006